NOTICE

The quality of this microform is heavily dependent upon the quality of the original thesis submitted for microfilming. Every effort has been made to ensure the highest quality of reproduction possible.

If pages are missing, contact the university which granted the degree.

Some pages may have indistinct print especially if the original pages were typed with a poor typewriter ribbon or if the university sent us an inferior photocopy.

Reproduction in full or in part of this microform is governed by the Canadian Copyright Act, R.S.C. 1970, c. C-30, and subsequent amendments.

AVIS

La qualité de cette microforme dépend grandement de la qualité de la thèse soumise au microfilmage. Nous avons tout fait pour assurer une qualité supérieure de reproduction.

S'il manque des pages, veuillez communiquer avec l'université qui a conféré le grade.

La qualité d'impression de certaines pages peut laisser à désirer, surtout si les pages originales ont été dactylographiées à l'aide d'un ruban usé ou si l'université nous a fait parvenir une photocopie de qualité inférieure.

La reproduction, même partielle, de cette microforme est soumise à la Loi canadienne sur le droit d'auteur, SRC 1970, c. C-30, et ses amendements subséquents.

Canada

The Development of a Model-based Robotic Controller Using the Transputer Technology

Fujun Zhao

A Thesis

in

The Department

of

Mechanical Engineering

Presented in partial fulfilment of the Requirements

for the Degree of Master of Applied Science at

Concordia University

Montreal, Quebec, Canada

February 1995

ISBN 0-612-01360-X

Abstract

The Development of a Model-based Robotic Controller Using the Transputer Technology

Fujun Zhao

This thesis describes the development of a model-based robotic controller for industrial manipulators at the Centre for Industrial Control, Concordia University. Using the Transputer technology, with its parallel processing capabilities, sampling rate around 800 Hz is achieved for the real-time control of a 6-DOF Puma robot based on dynamics, which has never been demonstrated. The developed controller, specifies the task of the robot in terms of the end effector of the robot in Cartesian space, compensates the dynamics of the robot in real time, is able to provide industrial robots with a high speed, high accuracy trajectory tracking for non-repetitive motions in the free work space of the robot. Important aspects of the development, including evaluation of the designed hierarchy by simulation modelling, controller hierarchy design and implementation using the Transputer technology, real time experimental results are covered in a great detail. Extensive test is performed to investigate the effect of the sampling rate and the velocity of the desired trajectory on the tracking performance of the robot. The former investigation provides a valuable guidance for practical robotic controller design based on dynamics, while the latter investigation shows that the developed controller is quite robust, showing little performance degradation with the speed variation of the desired trajectory.

Acknowledgements

The author dedicates this thesis to the thesis supervisor Prof. R.M.H.Cheng for his invaluable guidance and consistent encouragement through out the course of this research and looks forward for his guidance in all future endeavour.

The author would like to express his appreciation to Dr. S.LeQuoc for the financial support and his encouragement.

The author is indebted to Mr. G. Huard, whose enthusiasm to build and make the interface cards and power stage work played a significant role in the development of this project.

The author would like to express his sincere thanks to Mr. S. Kabra for the useful discussions with the author, to the Machine Shop of the Department of Mechanical Engineering, Concordia University for building of the experiment set up.

My dedication to my wife, my son, and other members of the family for their continued support and encouragement all through these years.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Overview

M. Brady in 1989 [1] listed the eight more difficult challenges for modern control theory to be applied in advanced robotics. Complex rapidly changing dynamics and non-linearity are two of them. Indeed, the dynamic model of a typical industrial robot of N degrees of freedom is described by a set of N coupled, highly non-linear second order differential equations, which requires a non-linear control scheme to compensate for the dominant dynamics of the robot for high performance and high speed practical applications.

Due to the relatively simple implementation, the classical independent joint PID control using the position and velocity feedback loops dominates the controller design of industrial robot that are available commercially. It works well in practice for low speed and low-performance operations. For instance, considering a Puma 560 robot, the maximum linear velocity of the end effector could reach 1 m/s. With the VAL controller (PID control basically) provided by the vendor, only half of this speed is achievable for practical operations [2]. Also, the maximum position error is possible to exceeds 3 degrees in joint space [3].

Model-based control schemes, such as computed torque control, is proposed to overcome the above mentioned problems. The motivation of computed torque methods,

which are also named as inverse dynamic techniques, and dynamic control, is to design a non-linear control input which transforms the initial non-linear system into a decoupled time-invariant linear system. And then classical linear feedback loops could be applied in the joint space of the robot. It has been being worked on for many years. However, most of the research is based on only simulation modelling. Real-time experimental evaluation lags behind by a large margin for following reasons:

1) Rarely industrial robot allows the torque control of their actuators.

2) Model-based robot control requires much more computation power requirement for the inverse dynamics solution of the robot in real time.

The above two reasons require to redevelop the hardware system of the controller provided by the vendor. Hence the software system must also be redeveloped.

3) As the speed of the dc servo motor which is used by most industrial robots as actuators, is directly related to the voltage between the two terminals of the armature circuitry, and the control of the robot is done in the joint space, the control of the robot could be achieved by a typical IC motion controller like HP1000 and LM 628/629 motion controller [4]. As far as the model-based control of the robot is concerned, it is absolutely impossible to get a commercially available typical IC chip which functions as a torque controller due to the vast diversity of the industrial robots or any other controlled mechanical systems. This makes the real-time implementation of the model-based robot control much more complicated than that of the traditional independent PID control.

1.2 Objectives

The objective of this research is to develop a controller based on dynamics for high speed and high performance robotic operations, to provide a platform for the realization of a model-based control scheme for the real-time performance evaluation. Important issues for the practical robotic controller design, such as the effect of the sampling rate on the tracking performance, effect of the operation speed on the tracking performance, will be investigated based on extensive experimental results. The motivation is to make an effort towards making the model-based controller design as the routine practice for advanced industrial robots.

A Puma 560 robot arm is selected as the test rig due to the fact that Puma 560 robot arm is available in our laboratory, and it is a typical geared industrial robot, and that parameters related to the dynamical model have been identified.

1.3 Thesis Organisation

Chapter 2 consists of two main parts. The first part is a brief description and discussion of various control schemes used by industrial robots, or proposed by researchers to solve trajectory tracking problems. This discussion verifies the necessity and imperativity to utilize dynamics control on the controller design for advanced high speed and high performance robots. Part two is a description of research achievements on model-based robotic control with emphases on real-time evaluation.

A concept design of the controller hierarchy for realization of model-based control schemes is proposed in Chapter 3. It is evaluated by simulation modelling. Important

issues for control, such as the choice of the controller gains, the effect of the sampling rate on the trajectory tracking performance, effect of the speed of the trajectory on the tracking performance is also investigated based on the simulation results.

Chapter 4 presents in great detail the design and implementation of the controller using the transputer technology, including the design of the controller hierarchy architecture, hardware and software architecture, and the implementation of the computed torque control on a Puma 560 robot arm.

The experimental results of the developed controller, using computed torque control scheme, are given in Chapter 5, both in the joint space and cartesian space, at different speed ranging from low to an extremely high close to the maximum speed of the mechanical design or actuators of the robot arm. The results of the computed torque control is compared with the traditional PID control implemented in [6].

Summary, conclusions and the recommendations for future work are presented in Chapter 6.

Enclosed in Appendix A is the real-time implementation of the computed torque control on a simple mechanical set up, while a dc servo motor with a gear head is used to simulate the driving system of a joint of a geared manipulator, two inertia disks attached to the shaft of the gear box is used to simulate one link of a manipulator. Considerable knowledge has been gained from this work for the real-time implementation of computed torque control on geared industrial robots. The key microprocessors, Transputers for building the parallel processing network of the controller, and developing concurrent applications on transputers is included in Appendix B.

Chapter 2

A Survey on Industrial Robot Control

2.1 Introduction

This chapter consists of two main parts. The first part is a summary and discussion of various control schemes currently used and/or proposed to study for solving trajectory tracking problems with a emphasis on the model-based robotic control. The study shows that the model-based control scheme is the proper control scheme for solving the control problem on robotics. It leads to the adoption of a model-based control scheme in the controller design in order to achieve the objective described in Chapter 1. Part two is a summary and/or survey on the research achievement on the application of model-based control in robotics. Although it has been proposed decades ago, and has been worked on for quite some time, most of the work is based only on software simulation [31], the solution of inverse dynamics aims at minimizing the numbers of the arithmetic operations [32], [45], [74] - [76], and scheduling algorithms using parallel architectures driven mainly by the need for fast computation of complex inverse dynamics [7] - [10]. Real-time evaluation lags behind theoretical study by a large margin because of the problems and difficulties described in the previous chapter. Real-time evaluation of model-based control on industrial robots is very imperative if the model-based control is claimed to be taken seriously in the practical controller design for advanced industrial robots for high speed and high performance operations.

2.2 Control Schemes for Solving Trajectory Tracking Problems

2.2.1 Manipulator Control Problem Statement

A robot task is naturally specified in terms of its end effector in cartesian space. The measured variables for feedback purposes are joint displacements and velocities. Usually the cartesian trajectory are transformed into corresponding points in joint space by the kinematics of the robot. These joints space points form the corresponding joint trajectories and are then used as the reference input points for the control of each joint. Hence the control is done entirely at the joint level, although the goal is the position and orientation of the end effector of the robot in cartesian space.

The manipulator control problem involves the computation of the joint torques/forces that must be applied in order to make the manipulator track the desired trajectory. The dynamic model of a manipulator is described as a set of highly nonlinear and coupled differential equations. The dynamic model of a 6-DOF manipulator has the form [11]:

$$\tau = M(\theta)\ddot{\theta} + V(\theta,\dot{\theta}) + G(\theta) + F(\theta,\dot{\theta}) \qquad (2.1)$$

where:

τ is the 6x1 vector of joint torques,

$M(\theta)$ is the 6x6 inertia matrix of the manipulator,

$V(\theta,\dot{\theta})$ is the 6x1 vector of Coriolis and centrifugal forces,

$G(\theta)$ is the 6x1 vector of gravity forces, and

$F(\theta,\dot{\theta})$ is the 6x1 vector of the friction model.

2.2.2 Independent Joint PID Control

The independent joint PID control scheme (PID) is by far the most popular feed back controller used for industrial manipulators for the simple implementation. The servo torque is given by:

$$\tau = \ddot{\theta}_d + K_v\dot{e} + K_p e + K_i \int e\,dt \tag{2.2}$$

where $e = \theta_d - \theta$. Since sometimes the values of $\ddot{\theta}_d$ may not be available, and the control law reduces to

$$\tau = K_v\dot{e} + K_p e + K_i \int e\,dt \tag{2.3}$$

The block diagram of the scheme is depicted in Figure 2.1.

The gains are typically tuned experimentally to be critically damped in a selected typical configuration of the manipulator. The PID control must essentially be viewed as an approximate method since the dynamic model of the robot is highly nonlinear and coupled. It doesn't consider any information on the system dynamics. The performance is heavily dependent on the pay load of the robot. Low speed of operation and large peak error of the end effector of the robot is the direct result. For instance, the original controller which is a PID controller, of the Puma 560 robot produces very small final errors because of the stiffness of the control, but allows peak errors to exceed three degrees [12]. It is the most commonly used control scheme by industrial robots today because of the relatively simple implementation.

2.2.3 Computed Torque Control

The motivation of the Computed Torque (CT) control scheme is to make use of the full dynamic model of the manipulator to compensate not only the effects of gravity but also Coriolis and centrifugal forces, friction and the manipulator inertia tensor in order to reduce or minimize the tracking peak error without adversely impacting on the steady state response . The nonlinear terms will be fedback [11]. The control torque τ is computed using the following equations:

$$\tau^{*} = \hat{M}(\theta)[\ddot{\theta}_d + K_v\dot{e} + K_p e] + \hat{V}(\theta,\dot{\theta}) + \hat{G}(\theta) + \hat{F}(\theta,\dot{\theta}) \qquad (2.4)$$

where K_v and K_p are 6x6 diagonal matrices. The feedback terms $\hat{V}(\theta,\dot{\theta})$, $\hat{G}(\theta)$, and $\hat{F}(\theta,\dot{\theta})$ are the nonlinear feedback torque terms that needs to be calculated fast enough by computer to make it an effective feedback control. The symbol "^" indicates that the estimated arm dynamics model is used in the computation. The computed torque control scheme is depicted in Figure 2.2.

Let ΔM, ΔV, ΔG and ΔF denote the errors in the estimates of the above robot arm parameters, then

$$\begin{aligned} M &= \hat{M} + \Delta M \\ V &= \hat{V} + \Delta V \\ G &= \hat{G} + \Delta G \\ F &= \hat{F} + \Delta F \end{aligned} \qquad (2.5)$$

The control law (2.4) in conjunction with the dynamic model (2.1) yields the following error equations [13]

$$\ddot{e} + (I - M^{-1}\Delta M)(K_v\dot{e} + K_p e) = M^{-1}(\Delta V + \Delta G + \Delta F + \Delta M\ddot{\theta}_d) \qquad (2.6)$$

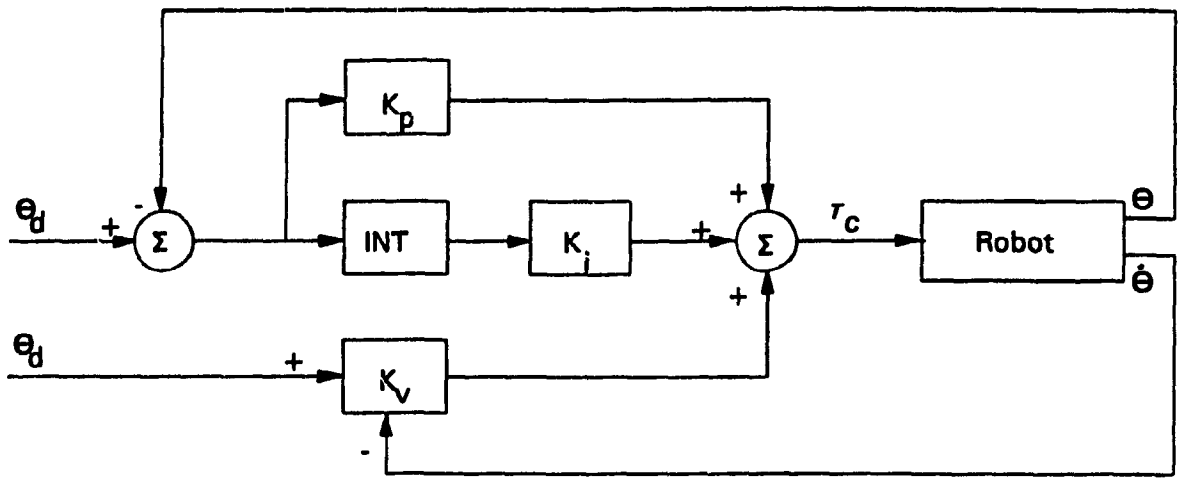


Fig. 2.1 Block Diagram of the PID Control Scheme

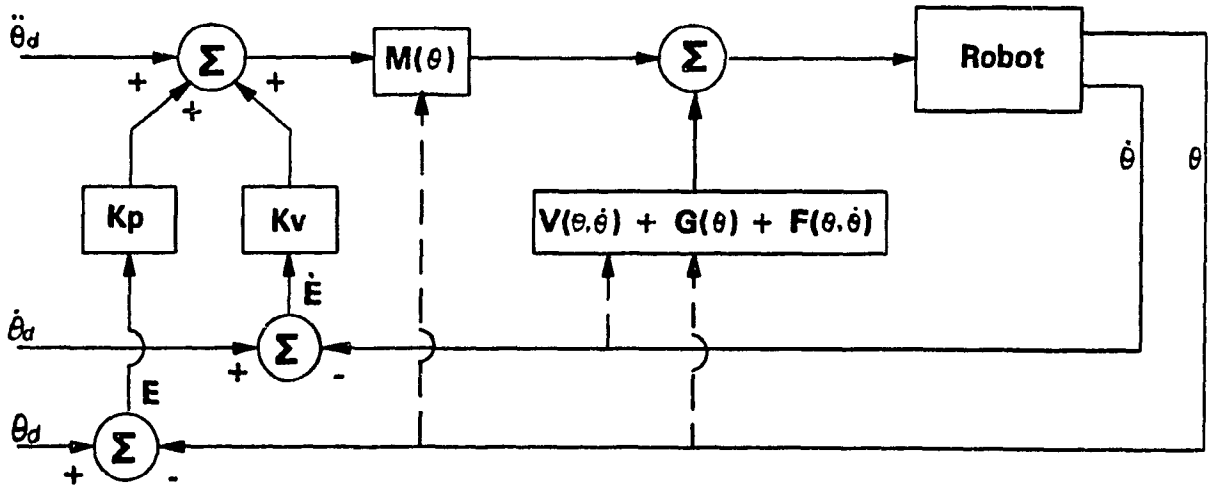


Fig. 2.2 Block Diagram of the Computed-torque Control Scheme

We could see clearly from the above equations that when the estimates of the model is exact, the error equations will reduce to a set of linear second order equations given by (2.7). And the resulting error feedback gain matrices, i. e. the nominal torque versus position error and the nominal torque versus the velocity error, are described by $\mathbf{MK}_p$, and $\mathbf{MK}_v$ respectively. Therefore the poles are fixed, and the feedback gains are state dependent and hence change along the trajectory.

$$\ddot{e} + K_v\dot{e} + K_p e = 0 \tag{2.7}$$

The above error equations are independent of the model of the robot, and thus the nonlinear terms are completely cancelled in this ideal case, since the gains matrices K_v and K_p are diagonal, the closed loop equations of motion are not only linear but also uncoupled from one another. If there is no noise and initial errors, the robot will follow the desired trajectory exactly. And if there is some noise or initial errors, the errors will be suppressed according to (2.7).

Obviously, the pay load can be always be taken into consideration in the dynamic model of the robot. Hence pay load independent trajectory tracking is achievable.

There are a number of practical drawbacks to the computed torque control method. The first, of course, is that the estimates of both the robot arm and the driving system parameters are never exact. Another deficiency which prevents complete cancellation of nonlinear effects is the observation that the model itself is never complete. For example, it dose not take into account such things as flexibility in the links and/or joint(harmonic drive system), gear backlash, effect caused by the uncertainty of the friction characteristics of the joint. In [14] it has proved that the stability of the CT

control is directly related to the structure of the dynamic model, not the errors in the estimates of the parameters appeared in the dynamic model. But a degradation of the trajectory tracking performance was noticed. Nevertheless, with the improvement of the measurement, it is possible to get an acceptable dynamic model in terms of the desired accuracy.

The requirement to compute the inverse dynamics of the robot on-line has been a bottleneck for years [15]. With the advanced computer technology today, such as transputer technology, with its parallel processing capability, low-cost, higher sampling rate than 800 Hz for real-time control could be achieved.

The effect of the velocity dependent Coriolis and Centrifugal term on the trajectory tracking performance of the manipulators has been a subject of controversy [41]. For some particular manipulators and /or operations, these terms could be ignored, then the control law is called, sometimes, the reduced computed torque control and is described by

$$\tau^* = \hat{M}(\theta)[\ddot{\theta}_d + K_v\dot{e} + K_p e] + \hat{V}(\theta,\dot{\theta}) + \hat{G}(\theta) + \hat{F}(\theta,\dot{\theta}) \qquad (2.8)$$

2.2.4 Feedforward Dynamic Compensation with PID Control

Feedforward Dynamic Compensation with PID control (FFDC/PID) is also called independent joint PID servo scheme with feedforward torque computation method. This scheme is based on the premise that the gross torque for trajectory tracking is provided by applying the joint torques computed from the inverse dynamic model. This feedforward signal is then augmented with the feedback signal derived from linear

independent joint controllers which are assumed to correct for the small deviations in trajectory tracking out of the mismatch in the dynamics of the model and the real arm. The goal of the dynamic compensations is to reduce the disturbances to the PD loop by linearizing and decoupling the coupled nonlinear manipulator dynamics. Feeding forward nominal torques performs a local linearization while the computed torque technique described above globally linearises system dynamics. Both methods allow improved trajectory tracking accuracy. The control torque is therefore:

$$\tau^* = \hat{M}(\theta_d)\ddot{\theta}_d + \hat{V}(\theta_d,\dot{\theta}_d) + \hat{G}(\theta_d) + \hat{F}(\theta_d,\dot{\theta}_d) + [K_v\dot{e} + K_p e] \qquad (2.9)$$

where the first four terms are the feedforward compensation torque, and the last term is the torque due to the feedback controller, and K_v, K_p are 6x6 diagonal gain matrices which are tuned in accordance to a typical robot position.

The control law (2.9) in conjunction with the dynamic model (2.1) yields the following error equations:

$$\ddot{e} + M^{-1}K_v\dot{e} + M^{-1}K_p e = 0 \qquad (2.10)$$

The resulting error feedback gain matrices are given by K_p and K_v respectively.

It can be seen that the FFDC/PID scheme has a nonlinear error dynamics. The feedback gains are fixed along the trajectory. Comparing the error equations of CT scheme with that of the FFDC/PID scheme, the former should provide a more accuracy in trajectory tracking performance than the latter. Most previous research showed that the performance of the FFDC/PID control scheme is compatible with CT control scheme.

2.2.5 Model-based Adaptive Control

In the discussion of the CT control scheme above, we mentioned that one of the deficiency of the CT control scheme is caused by the dismatch of the control torque signal expressed by equation (2.4) and the dynamic model of the robot expressed by equation (2.1). Thus the servo errors will be appear in this case. These error signals could be used to drive some adaptation scheme which attempts to update the values of the model parameters until the servo errors disappear. However, most industrial robots are driven indirectly, where the motor torque is amplified using a gear mechanism. Hence, the effect of inertial variations due to errors of the parameters in the dynamic model at the motor axis is reduced by $1/r^2$, where r is the gear ratio [16]. Therefore the model based adaptive control is not suitable for geared driven industrial robot.

Another approach is to use part of the dynamical information to adjust the servo operation to achieve better performance [17] - [19], without the on-line computation of the inverse dynamics. But this approach virtually made use of some simplifying assumptions.

2.3 Research Achievement in Robotic Control Based on Dynamics

Model-based robotic control was proposed in the early of 70's [20], and it has been realized as an advanced control scheme for high speed and high performance robots. However, most of the work are based only on the simulation modelling, very few real-time evaluation has been done until now. The work on real-time evaluation is limited to the evaluation in the joint space (some joints or a particular joint of the robot).

Real-time evaluation was made feasible on direct drive robot first time in 1986 in MIT Artificial Intelligent Lab [21], and was reported at the same time in Carnegie-Mellon University [22].

In MIT, computed torque control was implemented on the MIT Serial Link Direct Drive Arm having three links. In a direct drive arm, unlike a conventional mechanical arm that is driven through gears, chains, and lead screws, the joint axes are directly coupled to rotors of high-torque electric motors, and therefore no transmission mechanism is included between the motors and their loads. Because of this, the drive system has excellent features of no backlash, small friction, and high stiffness. So it contains fewer uncertain factors and no higher order delay, and the system can be identified accurately.

A series of experimental study was performed and reported in 1986 [21], 1987 [23], and 1988 [14] on the above mentioned robot. Because of the limitation of the computational power of the controller, sampling rate of only 133Hz was achieved, which is very low for a rapidly changing dynamic system like robot. Although the real-time experimental results showed that the computed torque control outperforms the traditional PID control when the same sampling rate was used for both control schemes, the real-time experimental results is not acceptable for practical applications as the peak position errors exceeds 4 degrees in joint space.

In Carnegie-Mellon University, model-based robot control was evaluated in real-time on the CMU DD Arm which also has three links. A high sampling rate of 500Hz was achieved. The peak position error for joints 1, 2, and 3 under the computed torque

control were reported as 0.022, 0.023, and 0.008 radians respectively, while the peak position error for joint 1, 2, and 3 under traditional PID control were reported as 0.036, 0.032 and 0.018 radians respectively. Hence it is verified that computed torque control scheme clearly outperforms the traditional PID control scheme.

Reduced computed torque control scheme was also implemented to test the effect of the velocity dependent terms $V(\theta,\dot{\theta})$ as it had been intuitively argued before that the effect of Coriolis and centrifugal forces becomes important only at high speeds. But the experiments showed that these effects introduce trajectory tracking errors even at small joint velocity.

The RFFDC/PID control scheme has been implemented on the robot to study the effect of the off-diagonal terms in the manipulator inertia matrix. The experiments clearly showed that by not incorporating the off-diagonal terms, in the RFFDC/PID control scheme, results in loss of accuracy in tracking when compared with FFDC/PID control scheme. Further theoretical analysis have been done on [24] that it is possible for the off-diagonal terms to completely dominate the computation of the control torques.

One of the principle objections cited against the computed torque control scheme is that it is very sensitive to modelling errors [25]. In order to address this issue, some experiments were done where the model of manipulator was changed drastically. The elements of the inertia matrix were changed by 20%. An increase in the tracking errors as a function of modelling error was noticed. No instability was found. The instability of the system is dependent on the structure of the manipulator inertia matrix and not on the modelling errors. Similar observation has been made together with theoretical

justifications in reference [26].

To provide some insight into the effect of sampling rates in robot control, two types of experiments were conducted in [28]. First, the performance of each control scheme as the sampling rate was changed was compared, and second, the relative performance of the CT and PID schemes at different sampling rates were compared. As the sampling rate is changed from 500 Hz to 200 Hz, the performance of the CT control deteriorates only marginally, the performance of PID control becomes completely unacceptable. This is because of the compensation for the nonlinear and coupling terms. Whereas it affects the PID scheme because the disturbance that is constituted by the nonlinear and the coupling terms is not rejected appreciably.

Both in the PID and the CT scheme, the controller gains(K_d, K_p and/or K_i) should be increased as the increase of the sampling frequency. The gain matrices K_p and K_d are a function of the sampling rate of the control system. the higher the sampling rate the larger the values of K_p and K_d can be chosen [27]. Therefor, the higher sampling rate dose not only imply improved performance but it also allows us to achieve high stiffness. It is desirable for a manipulator to have high stiffness so that the effect of unpredictable external disturbances on the trajectory tracking performance is significantly reduced.

Real-time experimental results of computed torque control of geared industrial robot was first reported in 1986 [29] - [30] using a Puma 560 robot as the test rig. Desired trajectory was not clear whether the task was planned in joint space or cartesian space. Because of the low sampling rate of the computed torque control (around 70 HZ), even the best control formulation was unable to reduce the tracking error sufficiently for

real-time gross motion implementation. However, this study recommended that both the Coriolis and the Centrifugal terms in the dynamical model should be ignored in the real-time control. Further study was performed and reported in 1987 [31], 1988 [32], [33] and 1989 [34] with a complete modelling of the dynamics of the driving systems of the robot. The tracking performance was improved with a peak position error of 0.75 degrees. Again, we are not convinced that the task was planned in the cartesian space. However, effects of the dynamic models on robot control was extensively studied which recommended that manipulators with high torque amplification (geared) drive systems

1) require accurate modelling of drive system dynamics.

2) reduce the computational complexity of the system dynamics by negating the impact of Coriolis and centrifugal terms,

3) reduce payload sensitivity of system dynamics, but not eliminate the requirement for accurate modelling of gravity and inertial coupling.

In 1990, real-time computed torque control scheme has been implemented on the first three joints of Puma 560 robot arm [35]. Based on their real-time parallel computation architecture for implementing the resolved Newton-Euler algorithm and their transputer based controller, a sampling period of 0.66 ms(about 1500 Hz) was accomplished. The whole system is composed of a host computer, a transputer network, the interface, and the manipulator. To assign a T800 Transputer to each joint of the robot for command torque generation, sampling period of 0.8 could be achieved by estimation.

The friction parameters were estimated by experiments. The trajectory was planed in joint space as follows: move from P0:(0,0,0) to P1:(-100,-10,70) in 2.5 seconds;

remain at this position for 1 second; move to P2:(100,70,-90) in 3.5 seconds; remain in this position for 1 second; return to P0:(0,0,0).

The tracking performance was evaluated by the maximum position error along the above mentioned trajectory. The experimental results showed that the performance of the computed torque control scheme outperforms that of the independent joint PID control scheme about 2 times. The performance of CT control scheme was a little better than that of the feedforward control scheme as shown in Table 1 bellow.

Table I Control performance (Max. Error in Degrees)

Control Law	Joint 1	Joint 2	Joint 3
PID	1.46	0.88	0.79
FFDC/PID	0.86	0.68	0.67
CT	0.87	0.32	0.51

Different control schemes based on dynamics of the robot including Nonlinear Feedback Controller, Independent Joint PD Servo with Feedforward Torque Controller, and the computed torque controller have been implemented on the first three joints of the Puma 560 robot arm with the last three joints frozen [36] in 1991. Trajectory planning was in task space. Two kinds of the tasks that have been experimented with are straight line and circular paths. Both involve formulation of an explicit time-based trajectory for the end effector position in three dimensional cartesian space using fifth order polynomial fit.

The Coriolis and Centrifugal forces are neglected due to the highly geared nature of the Puma 560 robot. The experiments suggest that neglecting them along the test

trajectories does not deteriorate performance. In fact, the computational time saved (increasing sampling frequency) by omission of these terms from the model seems to improve performance.

Friction Modelling: Friction is not incorporated in the robot model. It is compensated for by augmenting the voltage command to the controllers by friction parameter values that were experimentally determined.

A sampling frequency of 200 Hz was used.

The peak absolute errors in the algorithms involved dynamics are of the order of 2 millimetres (best reported peak absolute position error till date is about 0.5 mm).

All the control schemes based on dynamics of the robot perform better than the Independent Joint PID control scheme, while the tracking performance of all the control schemes based on dynamics are compatible.

The experimental results also indicate that trajectory tracking is highly sensitive to the gains. Higher gains call for higher sampling rates. Therefore, a high sampling rate is paramount for good tracking performance. There is a breakaway point at 200 Hz. After this point improvement in the tracking performance slows down. Based on the graph extrapolation that a sampling rate of 1000 Hz will decrease the absolute position error to about 0.5 mm.

The computed torque control scheme was implemented on the Robotics Research Corporation (RRC) K-1607 HP manipulator in 1992 [37].

The RRC K-1607 HP manipulator [38] is a 1.6 m long, seven axis unit with a 23 kg(51 lb) payload manipulator configured for factory and laboratory use. It is a 7-DOF manipulator with revolute joints. Each joint consists of an dc-electric servomotor and a harmonic drive reducer, which is unrivalled for compact, light, backlash-free torque

multiplication. But its application as a mechanism for directly driving joints in high performance manipulators is complicated by three factors intrinsic to the device:

It is relatively compliant and nonlinear in compliance.

It exhibits a two-per-input-revolution transmission error.

It has a considerable amount of internal friction.

The control architecture of RRC K-1607 HP manipulator could be categorized in general into two levels: the high level controller and the torque loop servo-controller.

The high level controller accepts cartesian tool points command from the user to execute the inverse kinematics transformations. Then according to the control law, a torque command will be generated based on the knowledge of the desired joint position, velocity, and the actual position, velocity of the joint. the actual joint position and angular velocity are measured by the resolvers and are converted to digital signals by an analog to digital conversion unit.

The low level controller is a torque loop servo-controller at each joint which regulates the actual torque output of the joint drive system. This torque loop servo-controller is used to overcome the problems resulting from the geared, flexible joint drive system. The torque is measured in the torque transducer with strain gauge. The torque loop functions to cause the drive to provide the command applied joint torque. However, it is not a easy job to include such a joint torque servo loop [42].

Attempt on the real-time evaluation of model-based control using a Puma 560 Arm as the test machine was also reported in [39] - [40], while the trajectory consists of a spline motion of joint 2, starting in the "Ready" position straight up, the joint being commanded to move through nearly 1 radian with gravity and then back against gravity, at maximum velocity, to the start position. All the other joints are commanded to remain stationary. It was reported that computed torque control outperforms the traditional PID control.

2.4 Summary

The discussion of section 2.2 showed that the CT control can provide the

industrial robot a pay load independent, high speed, high accuracy trajectory tracking for motions in the robot free work space. Although there are a number of practical drawbacks to the CT control, it is a control scheme with prosperous future. The final tracking errors is directly related to the stiffness of the system, so the inverse dynamics calculation must be performed fast enough to achieve a high sampling rate.

From the description in section 2.3, it can be seen that

1) Model-based control scheme outperforms traditional PID control if the same sampling rate is used for both control schemes.

2) As model-based control scheme requires much more computation power requirement than that of the traditional PID control, using same computer system, the traditional PID control scheme can achieve much higher sampling rate than that of the model-based control scheme. Therefore, using the maximum possible sampling rate limited by the same computation power requirement, traditional PID control scheme may outperforms the advanced model-based control scheme.

3) Some of the real-time evaluation of model-based control on robotics fails to perform better than the traditional PID control does because of the too low sampling rate (around 72Hz) which is much lower than that achieved by most robotic controller using PID control scheme (more than 1000Hz).

4) Very few real-time evaluation of model-based control scheme on industrial robots has been done till now. Real-time evaluation is very imperative to make model-based control scheme become a routine practice in the controller design for high speed and high performance advanced robots.

Chapter 3

Simulation Evaluation of a Model-based Robotic Controller

3.1 Introduction

The discussion in previous chapter indicates that using model-based control such as computed torque control with high sampling rate, fixed PD gains and adaptive feedforward dynamic compensation will provide industrial robots like Puma type robot with high speed, high accuracy trajectory tracking for non-repetitive motions in the robot free work space. Since the computed torque control scheme involves inverse dynamics calculation, controlling the actuated torques for each joint of the robot, and hardly any commercial industrial robot manipulators allow the control of the joint torque, both the hardware & software system need to be developed. Hence, a concept design of the controller hierarchy and the simulation evaluation of the design are needed.

This chapter is organized as follows: The concept design of the controller hierarchy is briefly described in section 3.2, dealing with its hierarchical architecture. Section 3.3 describes the methodology of the simulation, including the dynamic modelling, simulation, choosing of the controller gains and the verification of the correctness of the simulation. The reference trajectory, performance evaluation criteria, the effect of the sampling period on the trajectory tracking performance, and evaluation

results are presented in section 3.4. Presented finally is the summary.

3.2 Concept Design of a Model-based Robotic Controller

The Centre for Industrial Control has been using the transputers for quite some time to develop Integrated Robotic Workcell Controllers, and some of the progress has been reported [45,46]. This present project on designing and developing a model-based controller is part of the ongoing program.

A transputer [47] is a microprocessor with its own local memory and with provisions to link up to 4 other transputers. It could be used as a single *stand-alone* micro-processor system, or in a network to provide a high-performance parallel-processor system.

The hierarchical structure of the proposed controller is depicted in Figure 3.1. Its hardware construction consists of four layers: a PC-type computer functioning as the host of the transputer network and the operator console, a network of a cluster of transputers, a serial/parallel interface and the power stage. Task planning, trajectory conversion from cartesian space to joint space, control action generation according to CT control law are done by the transputer network of six T800 transputers. The estimates of the real joint position and speed is obtained from the shaft encoder attached to each joint of the robot. The control action will be updated once every 25 ms.

Via a serial/parallel interface, the transputer network is connected to the torque controller, one for each joint of the robot. Its function is to decode the measured joint position and velocity, send them to the control algorithm, receive the control torque from

the algorithm, convert the latter to an analog signal, and amplify it to actuate the joint motor, while all the time keeping the operation within the prescribed sampling period. The control torque for each joint will be updated once every 2 ms.

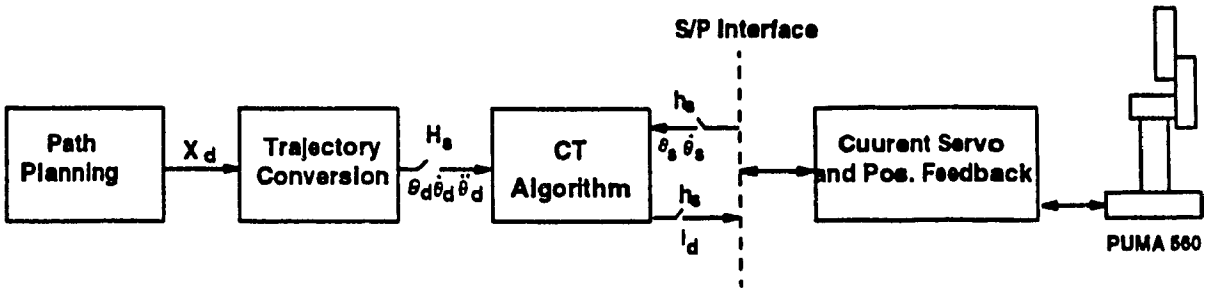


Fig. 3.1 Block Diagram of a Model-based Controller Hierarchy

3.3 Simulation Methodology

3.3.1 Dynamic Modelling and Simulation

The dynamical model of a manipulator is described as a set of highly nonlinear and coupled differential equations. The configuration space equation of the Puma 560 robot arm is described by the following set equations

$$\tau = A(\theta)[\ddot{\theta}] + B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] + G(\theta) \tag{3.1}$$

where:

$A(\theta)$ is the 6x6 kinematic energy matrix

$B(\theta)$ is the 6x15 matrix of Coriolis torques

$C(\theta)$ is the 6x6 matrix of centrifugal torque

$G(\theta)$ is the 6-vector of gravity torques

τ is the 6-vector of joint torques

θ is the 6-vector of joint positions

$$[\ddot{\theta}] = [\ddot{\theta}_1, \ddot{\theta}_2, \ddot{\theta}_3, \ddot{\theta}_4, \ddot{\theta}_5, \ddot{\theta}_6]$$

$$[\dot{\theta}\dot{\theta}] = [\dot{\theta}_1\dot{\theta}_2, \dot{\theta}_1\dot{\theta}_3, \ldots, \dot{\theta}_1\dot{\theta}_6, \dot{\theta}_2\dot{\theta}_3, \ldots, \dot{\theta}_5\dot{\theta}_6]$$

$$[\dot{\theta}^2] = [\dot{\theta}_1^2, \dot{\theta}_2^2, \dot{\theta}_3^2, \dot{\theta}_4^2, \dot{\theta}_5^2, \dot{\theta}_6^2]$$

The explicit dynamical model enclosed in reference [43] is used in the simulation. Friction at each joint of the robot is not taken into account.

The simulation requires solving the dynamic equations for acceleration.

$$[\ddot{\theta}] = A^{-1}[\tau^* - B(\theta)[\dot{\theta}\dot{\theta}] - C(\theta)[\dot{\theta}^2] - G(\theta)] \tag{3.2}$$

Given initial conditions on the motion of the manipulator in the form:

$$\begin{aligned} [\theta(0)] &= [\theta_0] \\ [\dot{\theta}(0)] &= [0] \end{aligned} \tag{3.3}$$

Equation (3.2) should be integrated numerically forward in time by steps of size Δt to get the simulated velocity and position iteratively, where for each iteration, equation (3.2) is computed to calculate $[\ddot{\theta}]$. And Δt should be sufficiently small that breaking continuous time into these small increments is a reasonable approximation. In this way, the position, velocity, and acceleration of the manipulator caused by a certain input torque function can be computed numerically.

3.3.2 Control Law

For the computed torque control, the control torque of the joints is calculated by the following equations:

$$\tau^* = A(\theta)(\ddot{\theta}_d + K_v\dot{E} + K_pE) + B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\dot{\theta}^2] + G(\theta) \qquad (3.4)$$

where the error E is defined by

$$E = [\theta_d] - [\theta] \qquad (3.5)$$

and K_v, K_p are 6x6 matrices of velocity and position error gains respectively. In the case that the model is exact, from equations (3.1) and (3.4), and choosing K_v, K_p to be diagonal gives the individual joint error equations:

$$\ddot{e}_i + k_{vi}\dot{e}_i + k_{pi}e_i = 0, \qquad i = 1,2,\ldots 6 \qquad (3.6)$$

The characteristic equation of the overall closed-loop system will be

$$s^2 + k_v s + k_p = 0 \qquad (3.7)$$

where k_v and k_v are two controller gains. The above equation could be expressed as

$$s^2 + 2\zeta\omega_n s + \omega_n^2 = 0 \qquad (3.8)$$

Compared equation (3.7) and (3.8), we get

$$\begin{aligned} k_p &= \omega_n^2 \\ k_v &= 2\zeta\omega_n \end{aligned} \qquad (3.9)$$

where ζ and ω_n specify the damping ratio and natural frequency for the desired

response. Selecting $\zeta = 1$ to get a critically damped system. then k_p and k_v should be determined to satisfy $k_v = 2\sqrt{k_p}$.

3.3.2.1 Choosing Controller Gains

Obviously, the higher the controller gains, the higher the stiffness of the control system. The high stiffness of the system implies accurate final trajectory tracking performance. But in practice, the velocity gain k_v is limited by the noise present in the velocity measurement.

One of the major assumption we have made in the model is that the gearing, shafts, bearing, and the driven link are not flexible. The argument for ignoring flexibility effects is that if the system is sufficiently stiff, the natural frequencies of these unmodeled resonances are very high and can be neglected compared to the influence of the dominant second-order poles that we have modeled.

Since we have chosen not to model structural flexibility in the system, we must be careful not to excite these resonances. [11] gives a recommendation that if the lowest structural resonance is ω_{res}, then we must limit our closed loop natural frequency according to

$$\omega_n \leq \frac{1}{2}\omega_{res} \tag{3.10}$$

Typically industrial manipulators have structural resonances in the range of 5 Hz to 25 Hz.

Real-time computed torque control scheme has been implemented on Puma 560 industrial robot arm in [32]. Considering the recommended value of natural frequency ω_n in [11], The natural frequency ω_n is selected as 15 for the first three joints of the robot, 20 for the last three joints, for the simulation evaluation. And the system poles are placed at -15 for the first three joints, and -20 for the last three joints respectively.

In real-time control, however, the upper limit of the velocity gain k_v could be determined experimentally. Set the position gain k_p to zero and increase the velocity gain k_v until the unmodeled high-frequency dynamics of the system is excited by the noise introduced in the velocity measurement. This value of k_v represents the maximum allowable velocity gain. Reference [28] chooses 80% of this value in order to obtain as high position gain as possible and still be within the stability limits with respect to the unmodeled high frequency dynamics. K_p then is computed to satisfy the critically damping condition.

3.3.2.2 Verification of the Correctness of the Dynamic Modelling

Reference [1] has presented the simulation modelling of the Puma 560 robot arm. The explicit dynamic model described in [43] was used. The two controller gains were chosen to be equal for all links with

$$\begin{aligned} \omega_n &= 160 \\ \zeta &= 0.59 \\ k_p &= 2.56x10^4 \\ k_v &= 1.89x10^2 \end{aligned} \tag{3.11}$$

The desired trajectory has the form of a staircase as shown in Figure 3.2. In order to test the correctness of the dynamic modelling and the simulation software, the same dynamical model, desired trajectory, controller gains were used in the simulation.

The control torque for joint 1 and the simulated position history are depicted in Figure 3.2 and Figure 3.3 respectively, which are very comparable with the results presented in [1]. Hence the correctness of the simulation software developed in this project is verified.

3.4 Controller Evaluation

3.4.1 Trajectory Selection and Evaluation Criteria

The task is specified in joint space, i.e. each joint of the robot is commanded to track a desired trajectory simultaneously. A fifth order polynomial was used to generate the desired trajectory for all joints. But the desired trajectory for joint 2 is on the opposite direction to the others as the joint range of joint 2 is limited by $-180 \leq \theta_2 \geq 43 \quad \wedge \quad 137 \leq \theta_2 \geq 180$. The joints are moved from (0, 0, 0, 0, 0, 0) to (100, 100, -100, 100, 100, 100) degrees in 1.0 second. The peak velocity is 187.5 deg/sec and the peak acceleration is 577 deg/sec^2. The desired trajectory position, velocity, and acceleration are depicted in Figure 3.4, 3.5 and 3.6 respectively.

For evaluating the performance of the robot control scheme, both the maximum tracking errors and the mean root square of the sum of the error squared are used. And both trajectory position errors and the trajectory velocity errors are considered.

3.4.2 Effect of the Sampling Rate on Performance

All the research results on model-based manipulator control showed that the higher the sampling rate, the better the performance could be achieved. But from the view point of real-time implementation of the computed torque control on industrial robot, say Puma 560 robot arm, we have to estimate that at least what sampling rate should be used in order to get a better tracking performance than the original controller of the Puma 560 robot before real-time implementing on the hardware. In order to know how the trajectory tracking performance varies with the sampling rate, we simulated the control system at eight different sampling rates.

Shown in Figures 3.7, 3.8 and 3.9 are the maximum tracking position error, the mean square root of the sum of the position error squared, and the final position error versus the sampling rate respectively. Figure 3.10 and Figure 3.11 show the maximum velocity error, the mean square root of the sum of the velocity error squared versus the sampling rate respectively.

From Figure 3.8, it can be seen clearly that even at a high sampling period of 20 ms the maximum position error is much smaller than the original controller of the Puma robot(the original Puma robot controller allows the maximum position error exceeds 3 degrees [32]). This showed the effectiveness of the dynamic compensation of the computed torque control scheme.

The minimal final position error of the original controller of the Puma robot is around 0.057 degrees [32]. From Figure 3.10, we know that this value is corresponding

to the sampling period of 10 ms(100 Hz). It should be mentioned that the above simulation result is based on the assumption that the dynamic model of the robot is exact, which is never the case, besides, the unmodeled joint dynamics such as gear backlash, flexibility uncertain friction which we never can be modeled exactly, will also affect the tracking performance [37]. Therefore, in real-time control, it is only possible that at a sampling period of less than 10 ms the computed torque control scheme could drive the final position error lower than the original controller of the Puma robot. Besides, from Figure 3.10 and 3.11, we know that the maximum velocity error, the mean square root of the sum of the velocity error squared decreases as the sampling period decreased. That is the main reason that the computed torque control scheme has been successfully implemented [39,44,35,36], and it is the main reason that have made some implementation less than ideal [21,32].

3.4.3 Performance Evaluation

In order to evaluate the performance of the designed controller, the desired trajectory is updated at the frequency of once every 25 ms, while the sampling period of the computed torque control remain fixed at 2 ms. The results are very similar to that obtained in section 3.4.2. The expected maximum tracking position error, the mean square root of the sum of the error squared, the maximum velocity error, the mean square root of the sum of the error squared are 7.5×10^{-3} radian, 2.7×10^{-3} radian, 0.025 rad/s., 0.012 rad/s respectively. But it should be mentioned that the above evaluation is based on the assumption that the dynamic model of the robot is exact. Besides, the

unmodeled joint drive dynamics will also affect the performance. Therefore the errors will be bigger in an actual system.

3.5 Summary

In this chapter, the concept design of a controller hierarchy utilizing computed torque control scheme has been outlined. And the proposed controller architecture has been evaluated by simulation modelling. The evaluation results showed that the peak position errors will be greatly reduced due to the dynamic compensation of the computed torque control scheme. The effect of the sampling rate on the trajectory tracking performance of the computed torque control has been investigated by extensive simulation experiments, which will provide an invaluable guidance for the first stage of the development.

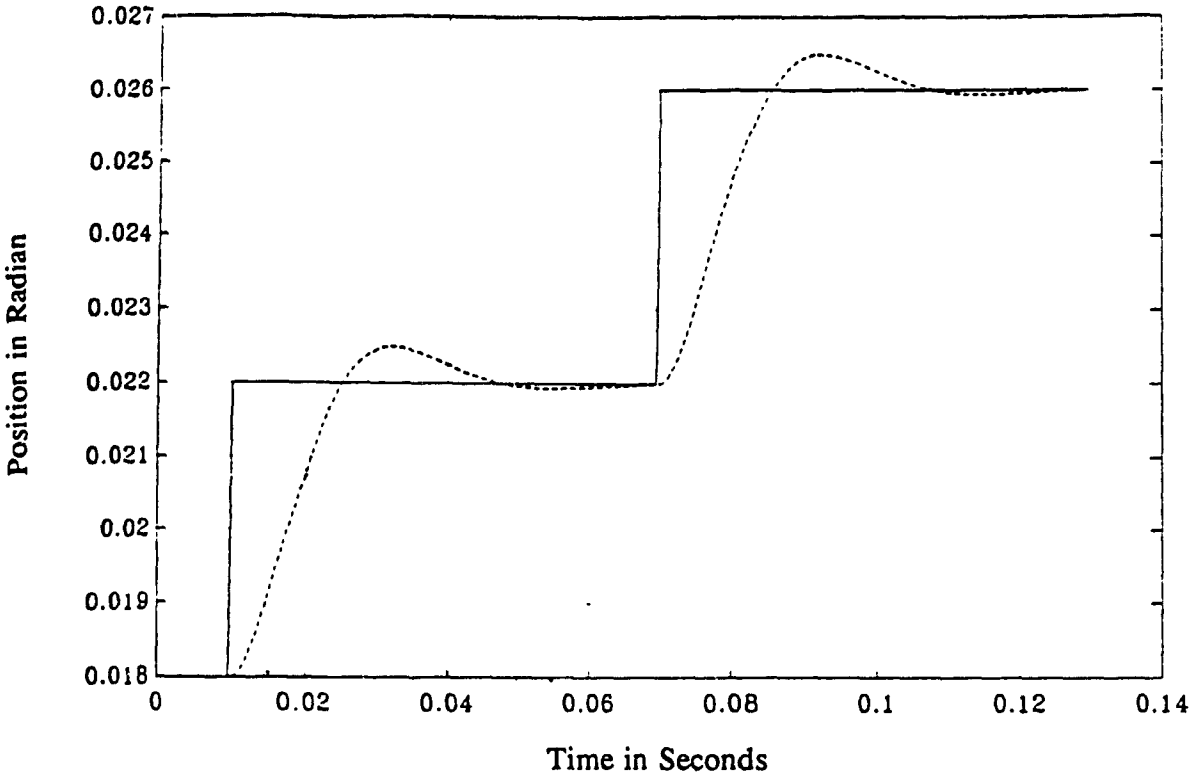


Fig. 3.2 Desired and Simulated Trajectory

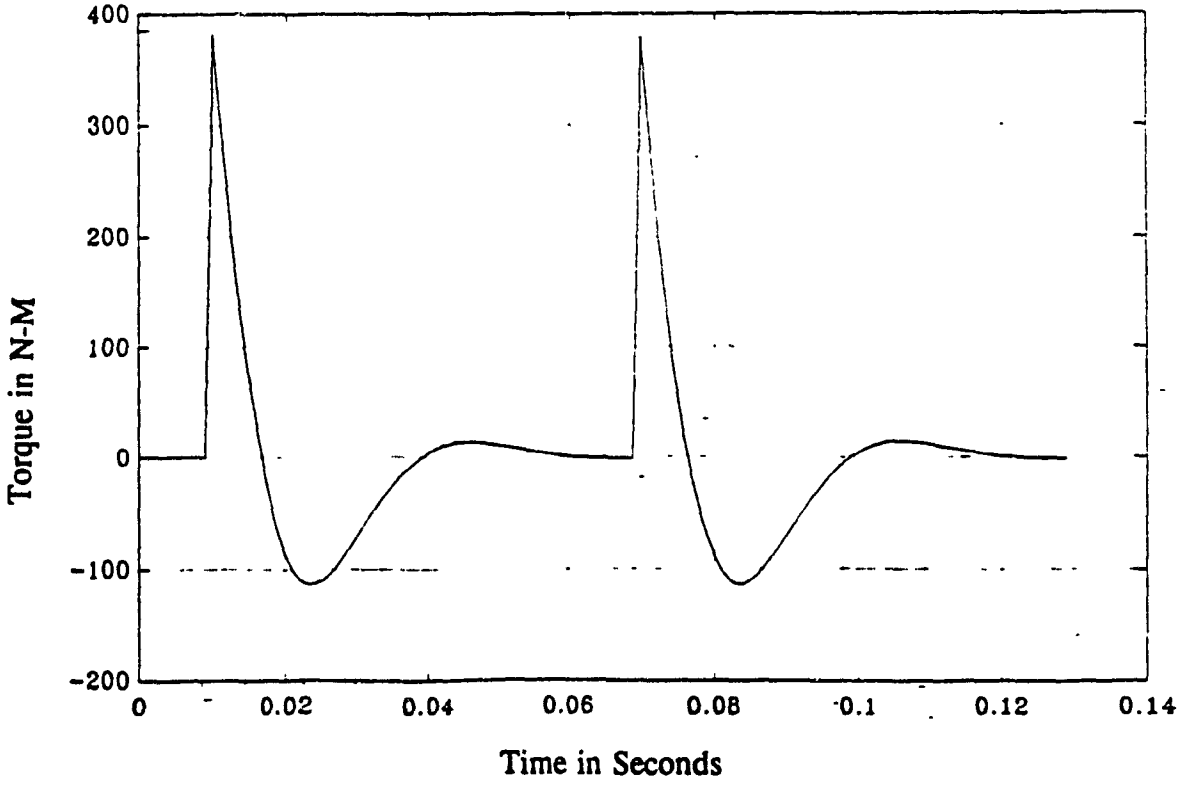


Fig. 3.3 Control Torque for Joint 1

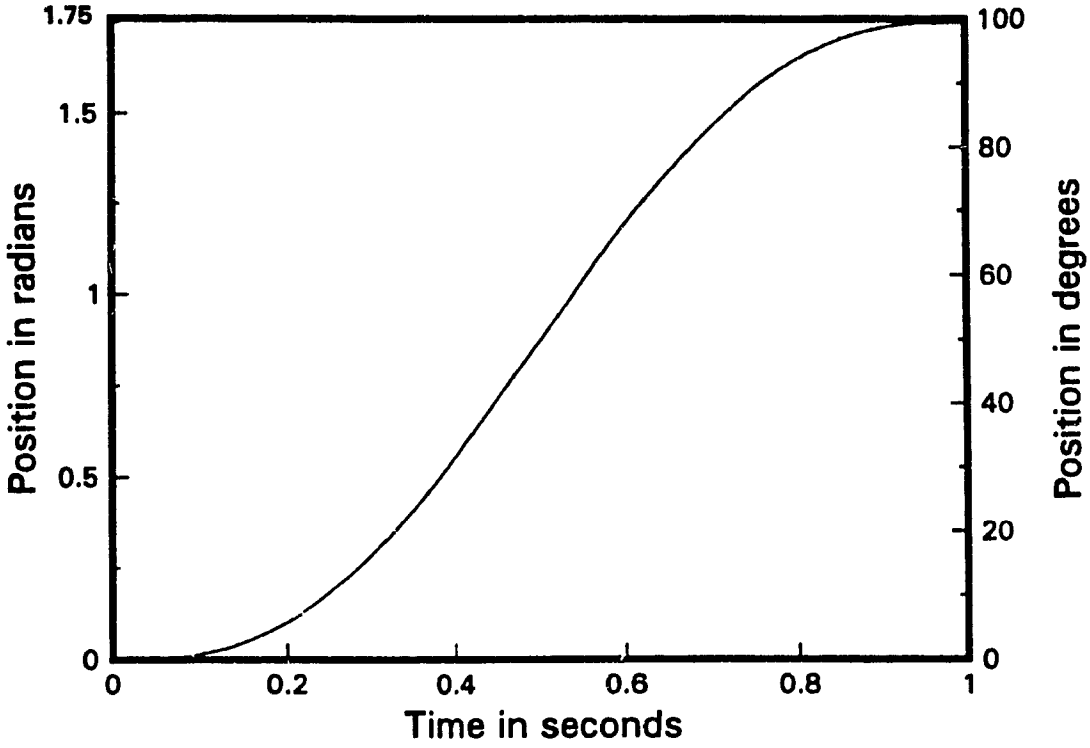


Fig. 3.4 Desired Trajectory for Simulation

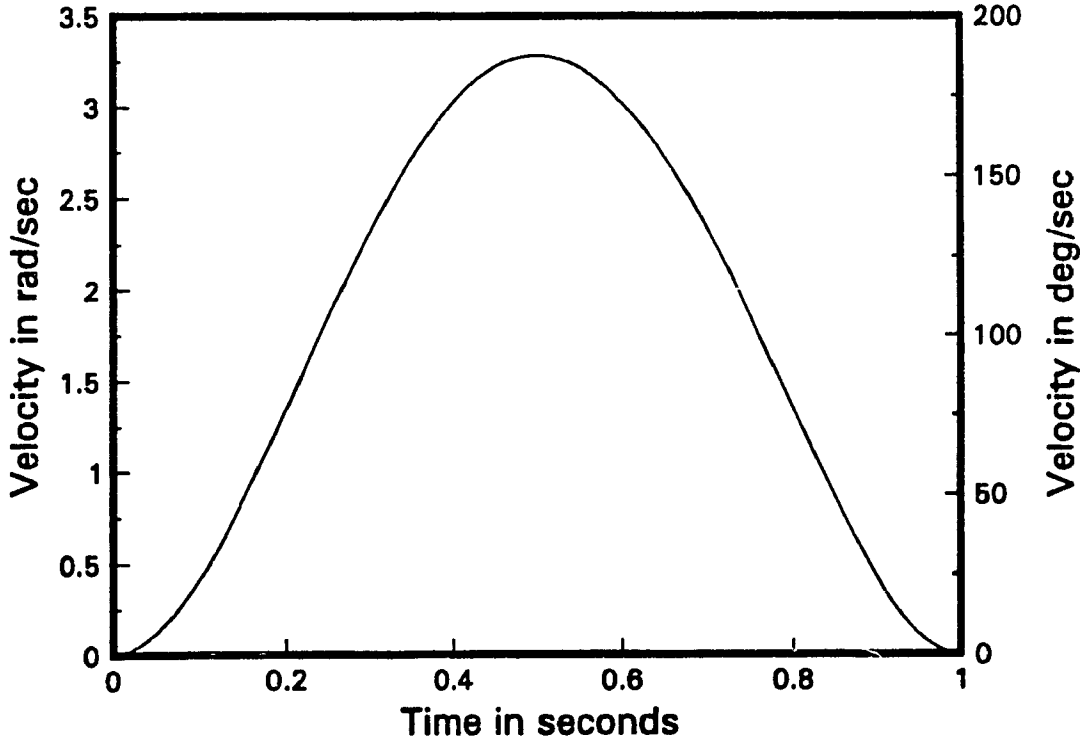


Fig. 3.5 Desired Velocity for Simulation

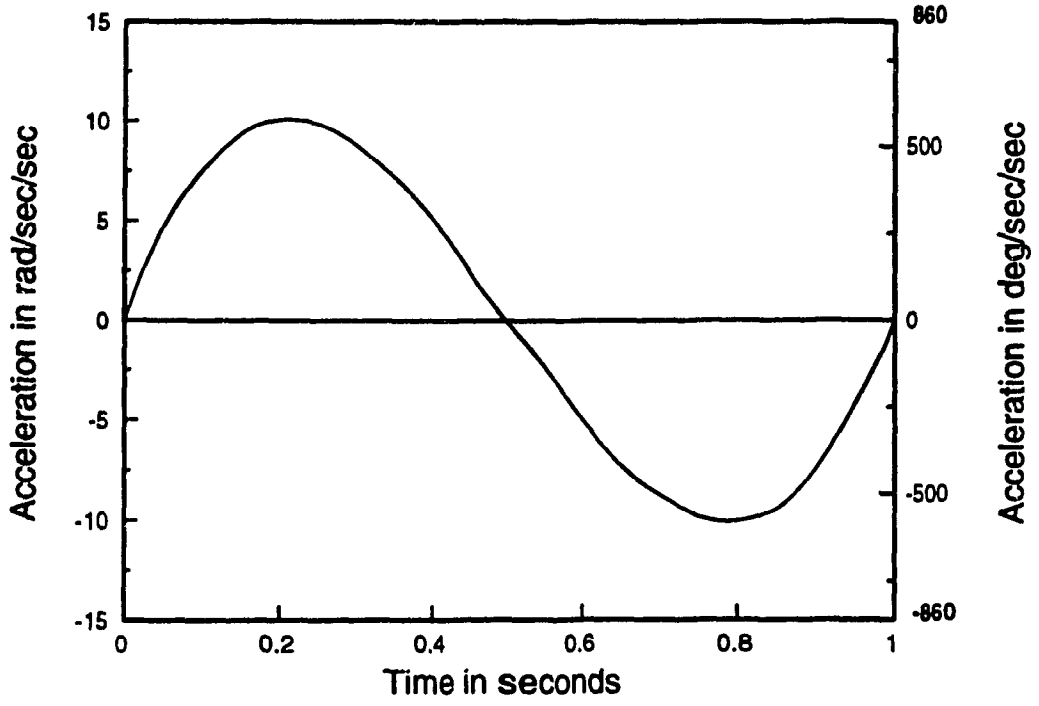


Fig. 3.6 Desired Acceleration for Simulation

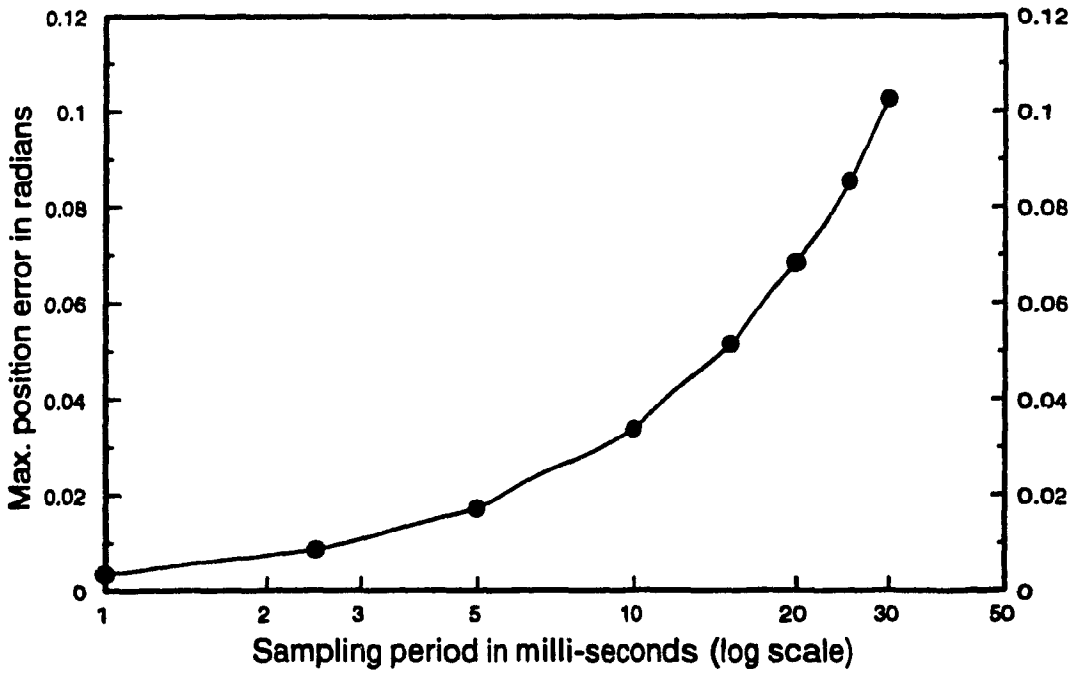


Fig. 3.7 Max. Position Error vs. Sampling period

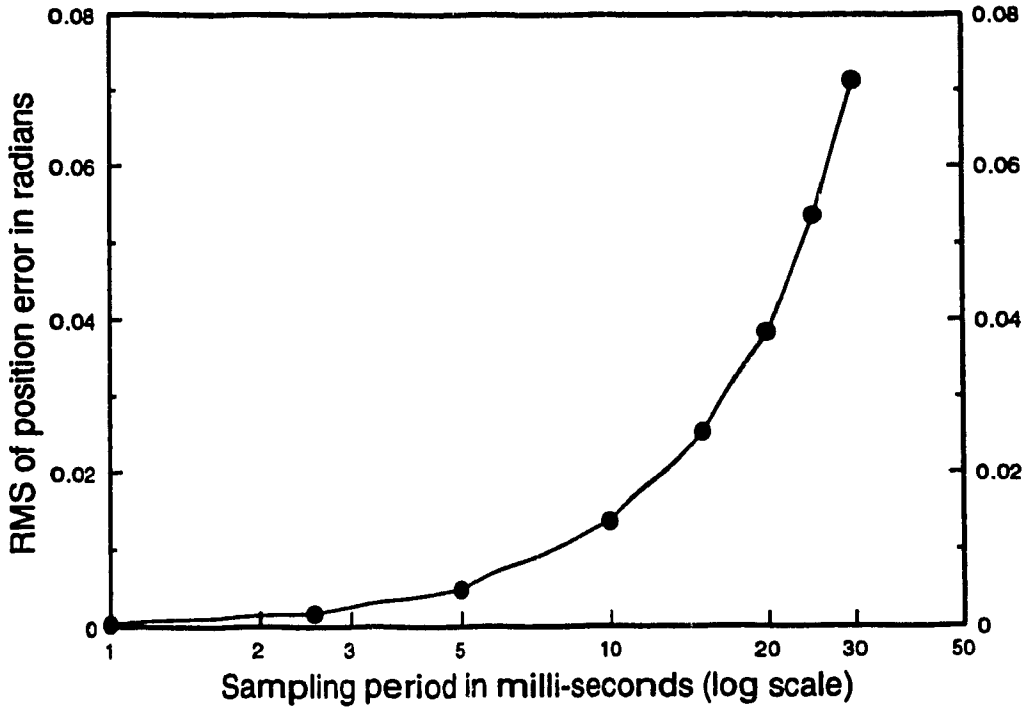


Fig. 3.8 RMS of Position Error vs. Sampling Period

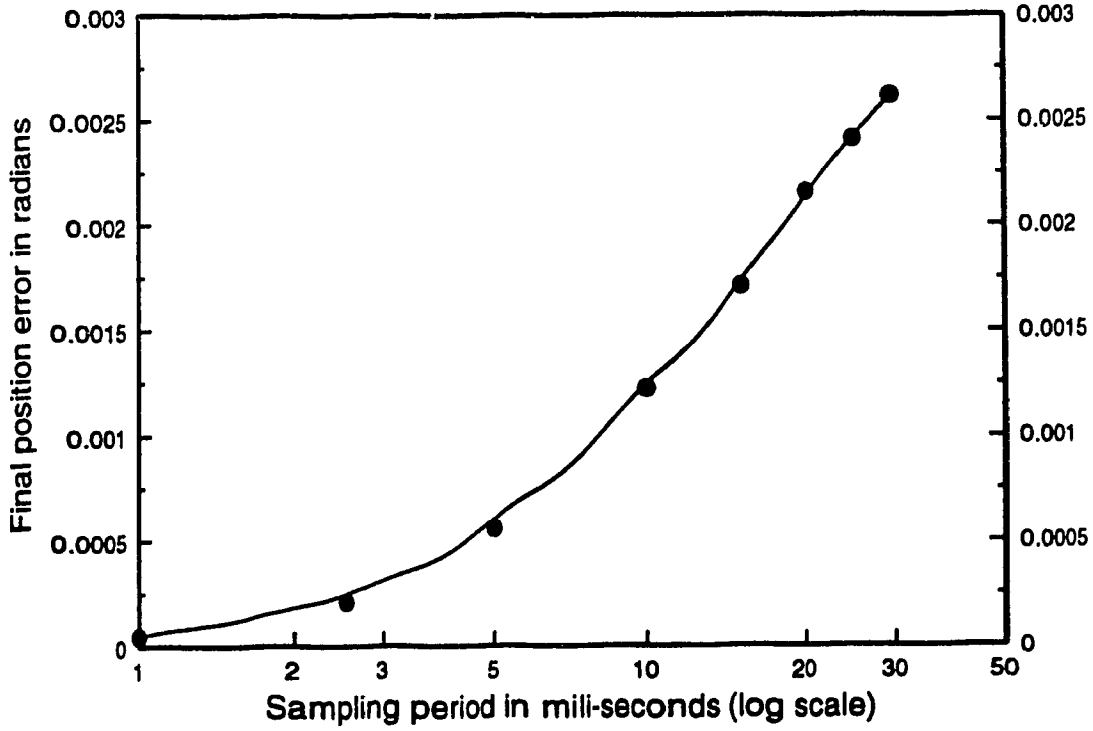


Fig. 3.9 Final Position Error vs Sampling Period

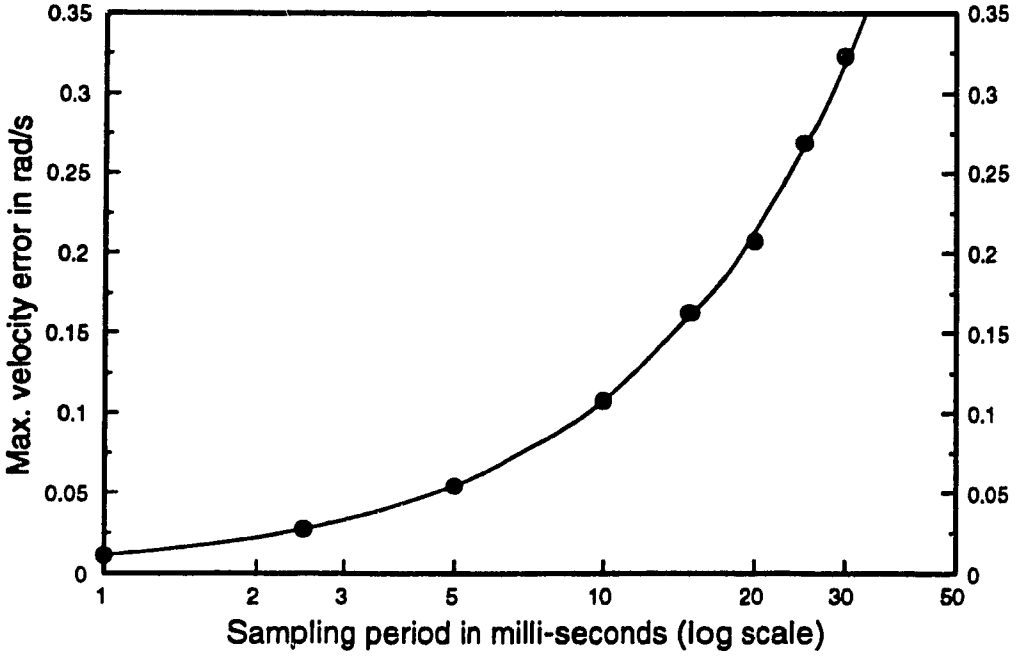


Fig. 3.10 Max. Velocity Error vs. Sampling Period

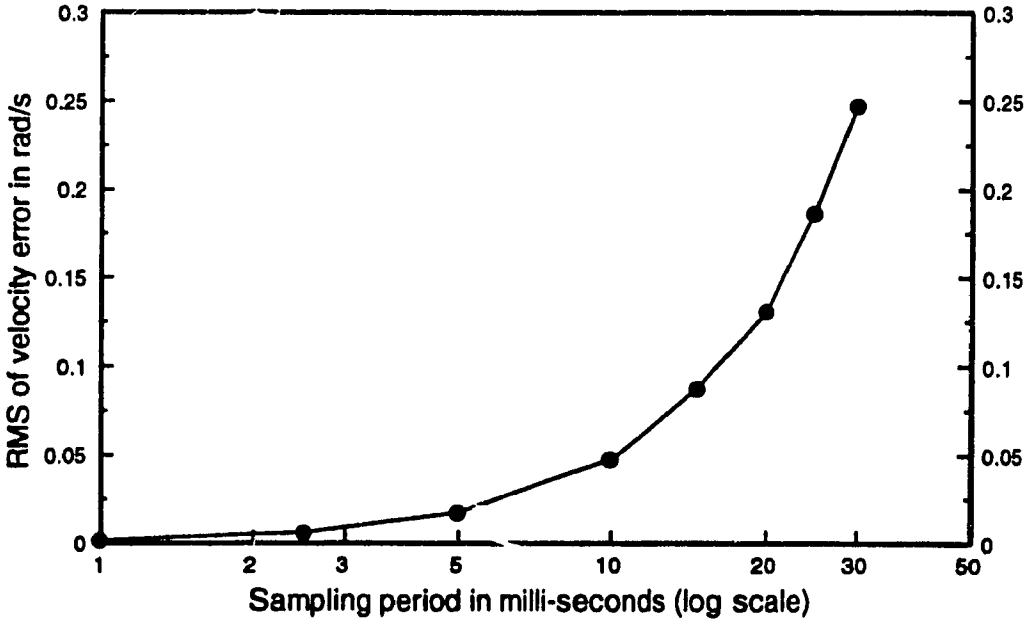


Fig. 3.11 RMS of Velocity Error vs. Sampling Period

Chapter 4

Design and Implementation

Using the Transputer Technology

4.1 Introduction

The discussion in Chapter 2 showed that model-based robot control scheme is a proper control scheme for solving the control problem on robotics simply because most industrial manipulators, like Puma 560 robot, is a highly nonlinear, coupled dynamic system. The research achievement on the robotic control based on dynamics showed that real-time evaluation of model-based control scheme is very imperative if it is claimed to be taken seriously in the practical controller design for advanced industrial robot. Simulation study of the computed torque control on Puma 560 robot leads us to a concept design of the controller hierarchy, which was also evaluated by simulation modelling. Real-time experiments on a simple mechanical load presented in Appendix B makes the implementation of the CT control on industrial robots stands on a more firm basis. This Chapter is dedicated to a comprehensive description of the design, architecture, and implementation of the model-based robot controller which resides on a IBM PC based transputer network.

This Chapter is organized as follows. Section 4.2 discusses in a considerable detail, the design, implementation of the controller hierarchy structure, the functions and

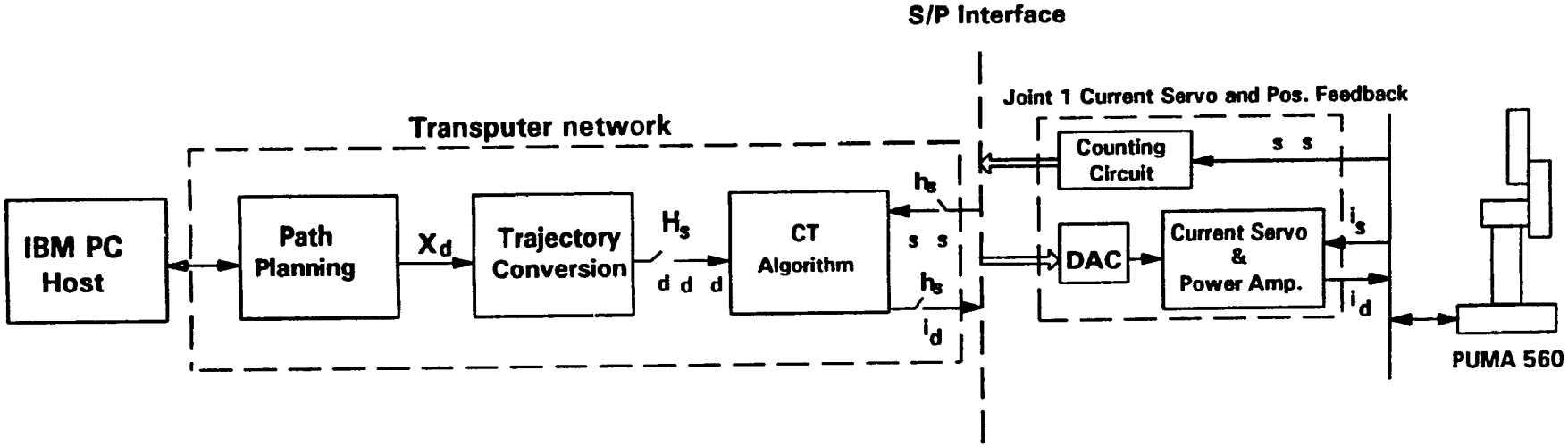
the principles of each layer in the control hierarchy with an emphases on the subject of the control torque generation according to the computed torque control algorithm. Section 4.3 presents the detail hardware system configuration. While section 4.4 gives a complete description of the software system architecture, the design and implementation of the controller.

4.2 Controller Hierarchy

The hierarchal structure of the controller, as shown in Figure 4.1, consists of five layers. At the top is the IBM PC, which acts as the host to the transputer network and as the OPERATOR console. Using a transputer link, it is connected to the second layer in hierarchy, the Cartesian Path Planner which depending on the execution phase,generates the desired cartesian path and gives suitable anchor points (in cartesian space) at each sampling instant to the next layer of the controller.

The Trajectory Converter forms the third layer of the controller hierarchy. The TC functions to convert the trajectory in terms of the end effector of the robot from cartesian space to the joint space by using inverse kinematics. Velocity and acceleration reference in joint space is numerically calculated as they are required in the formation of the control action by using the computed torque control scheme.

The CT control algorithm forms the fourth layer of the controller hierarchy. It functions to generate the control action for each of the six joints according to the CT control law described in Equation (3.4). The required joint actual position, and velocity vector are estimated from the readings of the motor shaft encoder. The explicit dynamical



Note: Hs = 28 ms

h_s= 1.25 ms

Fig. 4.1 CIC Model-based Robot Controller Hierarchy Architecture

Using a Puma 560 Arm as the Test Rig

model described in [43] is used. The insignificant terms are ignored to increase the CT control sampling rate without introducing tracking errors from the inaccuracy of the dynamical model.

The Serial/Parallel (S/P) interface and the power supply with a current feedback [48] forms the innermost control loop and the lowest layer in system design. The S/P interface functions to transfer data streams from the transputer serial link to parallel data streams for 8 bit devices like DAC, ADC, and the counter circuity for the position feedback from the shaft encoder, and vice versa. In the following sections, we consider the function of each level and its implementation using transputer technology in greater details.

4.2.1 Operator Console and Host

The IBM-PC serves the dual purpose of Operator Console and Host for the lower layers of the model-based robot controller. Using the PC as the host offers certain advantages. First, the processing power of the PC has and will increase substantially in the near future, accompanied by considerable decrease in cost. PC remains to be the most popular form of local intelligence in the industrial scenario and hence a wide variety of transducers, data acquisition systems and vision equipment are available as add-on PC peripherals. Further, the often quoted drawbacks of PC i.e. its limited graphics and multitasking capabilities are disappearing rapidly with the availability of windows for multitasking and add on computing engines for graphics processing.

The Transtech TMB08 PC Transputer Motherboard [49] occupies one slot of the

PC and supports all the lower layers of the controller. The TMB08 is a full length PC hosted Transputer Motherboard with space to plug up to ten Transputer Modules (refer Appendix B). The architecture of the motherboard supports the following:

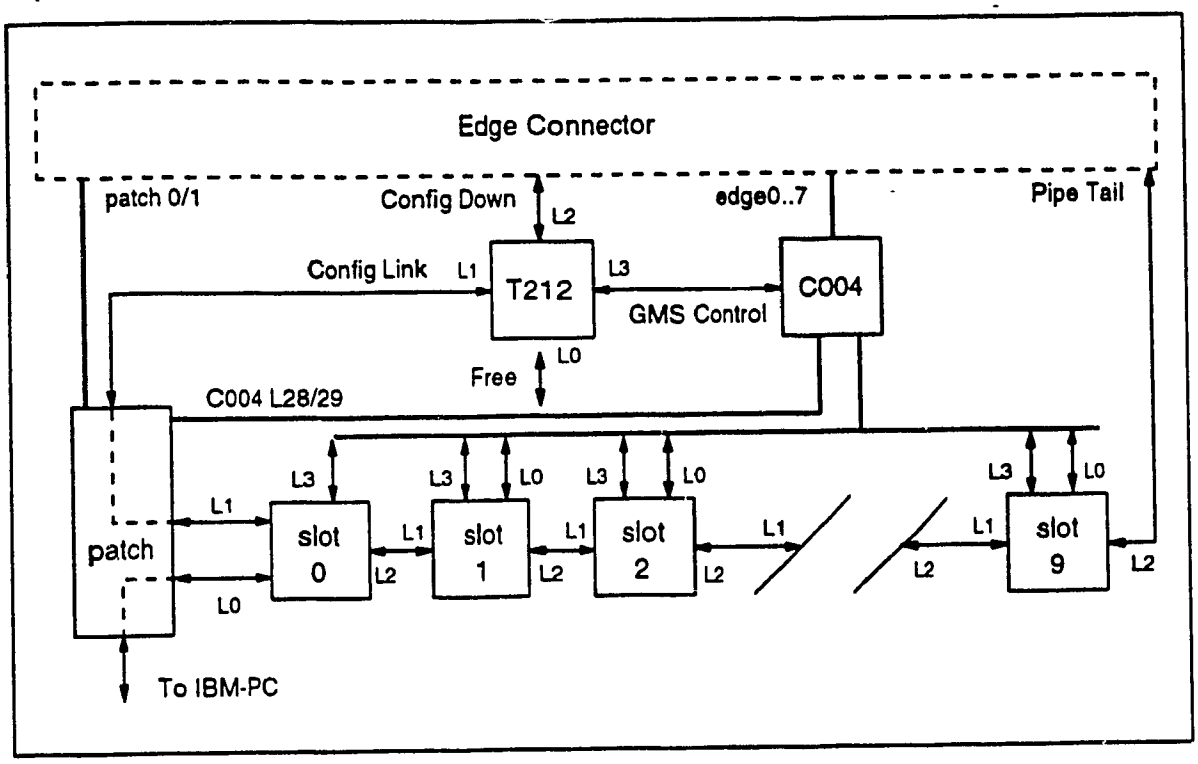


Fig. 4.2 The Default Transputer Interconnections in TMB08

- Build computing system consisting of any mix of TRAMS (size and type).
- Configure the transputers in any desired topology.
- Link a number of motherboards together to form a large network.

The important features of the motherboard architecture can be outlined as :

- The TRAM modules on the TMB08 are connected in a pipeline, using two links from each module as shown in Figure 4.2.
- The remaining links from each TRAM module can be configured by the

user through software programmable link switch IMS C004 present on board, for realising different network topology. The link switch is controlled by the on-board T212 transputer.

- The TRAM should be reset before a user program can be down loaded and executed by it. In the controller, reset and other control signals for all TRAM subsystems are supported by the host PC.

The set-up configuration for the TMB08 in the Controller is:

- Board Address : #150
- PC DMA channel used : #1
- PC Interrupt used : #3

The user must ensure that no other PC peripheral uses the same DMA/Interrupt channel. If however there is a clash with another add-in card, it is possible to change them as described in [49].

The various tools of the parallel C programming environment (compiler, linker, debugger etc.) reside in the host PC and are invoked from it under the control of DOS operating system. The DOS provides standard I/O (input/output), file I/O and access to other PC peripherals for the Parallel C program executing on the transputer network. The special interface task called afserver, mentioned earlier in Appendix A, is provided by the vendor for this purpose, which runs on the host PC. This task provides access to the host computer for tasks running in the transputer system using one link of the root transputer. One major drawback of the current implementation of the 3L transputer file server is that it utilises a polling mechanism to detect transputer request for services

thereby destroying the independent processing power of the PC and reducing it to the mere status of a server. However, this problems has been resolved earlier by the development of a Interrupt Driven Transputer File Server [50], in which it takes the form of a Terminate and Stay Resident (TSR) program in the DOS and is waken-up by the transputer request. Though the current controller is not making use of this enhanced file server, it is expected to do so in the future and free PC to perform other functions like providing a graphical interface to the user. The host computer interface of the transputer i.e. PC - TMB08 hardware interface, is a separate issue and not covered here as it is transparent to the end user.

4.2.2 Cartesian Path Planning

As mentioned earlier, the Cartesian Path Planning (CPP) forms the second layer of hierarchy in the controller architecture. This block performs the functions listed below.

4.2.2.1 Cartesian Path Generation

One essential aspect of CPP function is to generate the cartesian path to be pursued by the end effector of the manipulator, and to send the desired point on the path to the next layer of the controller hierarchy at every sampling (T_c) instance. The desired path may be pre-determined from the phase of operation or may be generated by using cartesian space interpolation techniques, discussed in [45], on sensor identified selective points. The later is not covered in this thesis. After having determined the next desired

point on the cartesian path the CPP communicates a suitable anchor point (in world coordinates) to the Trajectory Converter.

4.2.2.2 Multiplexing Requests for Host Services

This important function of the CPP results from the fact that the transputer file server residing in the PC cannot be shared between multiple transputer tasks. This is a direct outcome of the transputer point to point communication architecture. As a result, only one transputer task can make use of host services which includes C standard I/O functions like printf, scanf etc. This is a strong limitation specially in the design phase where such statements provide a strong support, and sometimes the only means, for debugging. The task filemux [51], which comes as an integral part of the 3L parallel C (version 2.2) and can accept requests from several processors and route them to the PC., allows several transputer tasks to share a single file server running on the host and is mapped on one of the processors of the CPP cluster.

To conclude, the CPP establishes a two-way channel between the host and the lower layers of Controller and makes the manipulator to perform some productive task. To perform this it makes use of TMB08 - PC hardware interface, software interface and the on-board link switch with switch controller.

4.2.3 Trajectory Conversion

As the task of a robot manipulator is typically defined in cartesian space in terms of the robot end effector, the control action is performed in joint space, the trajectory

must be transformed from the cartesian space to the corresponding trajectory in joint space. As shown in Figure 4.1, the inputs to the Trajectory Conversion (TC) are the spatial position vector as calculated in the path planing stage. The Trajectory Conversion transforms this vector into desired joint positions which are subsequently used as reference set points in the lower level CT control algorithm. The required joint velocity and acceleration vectors are numerically estimated. The transformation, which in the least requires the solution of inverse kinematic problem, is performed in each sampling interval T_c. There are two ways, as discussed below, to convert the trajectory from the cartesian space in terms of the end effector of the robot to the corresponding trajectory in joint space.

In an absolute convertor, using the inverse kinematic algorithm, the position vector **r** is transformed into the absolute joint angles **q**. At each sampling interval k, the **q(k)** vector calculated is sent as new reference position to the CT control algorithm for the control action generation. It is assumed that at time $t=(k+1)T_c$, all the joints will reach the absolute position **q(k)**. The required joint velocity and acceleration vector reference, are generated by taking

$$\dot{q}(k) = \frac{\delta q}{T_c}$$

$$\ddot{q}(k) = \frac{\delta \dot{q}}{T_c} \qquad (4.1)$$

where

$$\delta q = q(k) - q(k-1)$$

$$\delta \dot{q} = \dot{q}(k) - \dot{q}(k-1)$$

With Absolute Convertor, there is no reference position error at the intermediate

points. However, an error is induced between the intermediate set points because of constant speed profile over the sampling period. This error is dependent on the magnitude of δq and increases with longer sampling interval T_c or higher speed. Additional errors occur because of truncations and approximation of transcendental functions. However, since the calculation of the next reference position is independent, the **errors do not accumulate from one iteration to next**. Needless to say that the implementation of the absolute interpolator requires an explicit and unique solution to the inverse kinematic problem. This is not a problem for Puma type arm as such a solution exists.

An alternative is the Incremental Converter which transforms the spatial velocity vector into the corresponding joint speeds using the manipulator Jacobean. A stair case approximation of the speed is used as reference to the next layer of the controller hierarchy. Again, the reference speed remains constant for T_c period. The position reference is obtained by digital integration of speed i.e.,

$$q(k) = \sum_{j=1}^{k} \delta q(j)$$
$$\textit{where} \tag{4.2}$$
$$\delta q(j) = \dot{q}(j) T_c$$

The digital integration leads to an error between the required and reference speed for each joint. This error leads to an **accumulative error in path following**. Hence the absolute trajectory conversion technique is utilized in the implementation.

The kinematic analysis of the Puma type arm, using Denavit-Hartenberg (D-H) frame assignment, has been widely studied [52] - [57] and is not discussed in this thesis.

Parallelism in kinematic formulation of robot manipulators has been discussed by [58]-[59]. Forward and inverse kinematic formulation from [52] have been used in current work. The coupling in the wrist joints is discussed in [60, 61]. The arm related data for Puma 560 robot are given in Appendix E.

4.2.4 Control Torque Generation according to CT Algorithm

4.2.4.1 The Dynamic Model

CT control scheme, or model-based control which is recognized as one of the advanced manipulator control schemes, is the inverse dynamic problem, i. e., we are given a trajectory point vector, and we wish to find the required vector of joint torques of the manipulator. Many schemes of model-based control have been proposed, but most of them are limited within the stage of simulation studies [1], [62] - [64] as mentioned in Chapter 2. It is essentially an inverse dynamics problem to generate control action according to CT control law. Basically there are two ways to formulate the dynamical model of a robot manipulator [11], Iterative Newton-Euler dynamic formulation and Lagrangian formulation.

In the Iterative Newton-Euler dynamics formulation, the complete algorithm for computing joint torques from the motion of the joints is composed of two parts. First, link velocities and accelerations are iteratively computed from link 1 out to link n and the Newton-Euler equations are applied to each link. Second, forces and torques of interaction and joint actuator torques are computed recursively.

The Newton-Euler approach is based on the elementary dynamic formulas, and

on an analysis of forces and moments of constraint acting between the links. So it might be said to be a "force balance" approach to dynamics. As an alternative to the Newton-Euler method, the Lagrangian dynamics formulation should be referred to as an "energy-based" approach to dynamics. In the Lagrangian dynamics formulation, we start by developing an expression for the total kinetic energy and the potential energy of the manipulator. Then the Lagrangian dynamic formulation provides a means of deriving the dynamic equations of motion from a scalar function called the Lagrangian which is defined as the difference between the kinetic and potential energy of a mechanical system. Except for the two basic algorithms, i. e. Newton-Euler formulation and Lagrangian formulation, many other algorithms have been developed including Kane's formulation [64], or the model costumization approach [68] - [70]. However the computational requirements of these algorithms are still beyond the ability of commercially available microprocessors for high sample rate control. Hence the parallel processing approach becomes attractive.

A number of parallel processing schemes have been proposed, and a thorough discussion on them is given in [35]. Recently [9] has proposed a parallel processing schemes which has achieved a sample rate of 1000 Hz using three T800 transputers. And it is also limited on the simulation evaluation. [35] has proposed a resolved Newton-Euler Algorithm, which was evaluated using each T800 transputer for each joint on the Puma 560 arm as the test machine. Sample period of 0.66 ms was achieved. It resolves the process of inverse dynamic computations into subprocess that have potential parallelism. All sub-processes are given equally distributed computational burdens, and thus nearly

maximum concurrency is achieved.

Real-time implementation of the CT control scheme on industrial robots is only found on Puma 560 robot mainly because that the inertia dynamics has been extensively studied [9]. The dynamic parameters used in [35] is said to be obtained from the original design data by using a computer aided design system, but no detail was given. In this development, the explicit dynamical model derived in [43] is used. Parameters of the motor dynamics are taken from [73].

The dynamic model used in this implementation is presented by the following equation. It in fact is the explicit dynamic model with the joint friction torque vector.

$$A(\theta)[\ddot{\theta}] + B(\theta)[\dot{\theta}\dot{\theta}] + C(\theta)[\theta^2] + G(\theta) + \tau_f(\dot{\theta}, \tau_m) = \tau \tag{4.3}$$

The friction torque is described [33] by

$$\tau_f(\dot{\theta}\tau_m) = \begin{cases} \tau_c sgn(\dot{\theta}), & (|\dot{\theta}|) > d \\ \tau_c sgn(\tau_m), & (|\dot{\theta}|) \leq d \end{cases} \tag{4.4}$$

where d is recommended as 0.01 rad/s, and τ_c values the 90% of the stiction values. The dynamic parameters of the arm and the dynamic parameters of the driving systems are presented in Table 1 and Table 2 respectively in Appendix F.

4.2.4.2 Computed Torque Control

The control action is calculated by

$$\tau^* = \hat{A}(\theta)[\ddot{\theta}^*] + \hat{B}(\theta)[\dot{\theta}\dot{\theta}] + \hat{C}(\theta)[\theta^2] + \hat{G}(\theta) + \hat{\tau}_f(\dot{\theta}, \tau_m)$$

Where

$$\ddot{\theta}^* = \ddot{\theta}_d + K_d(\dot{\theta}_d - \dot{\theta}) + K_p(\theta_d - \theta) \tag{4.5}$$

Hence

$$I = \frac{\tau^*}{k_\tau}$$

where K_d and K_p are 6x6 diagonal matrices the estimates of which has been discussed in Chapter 3. The feedback terms are the nonlinear feedback torque terms that needs to be calculated fast enough by computer to make it an effective feedback control. And "^" indicates that the estimated arm dynamics model is used in the computation.

It should be mentioned that the computed torque control law described above could be replaced by other model-based control schemes, such as model reference control scheme, with little effort.

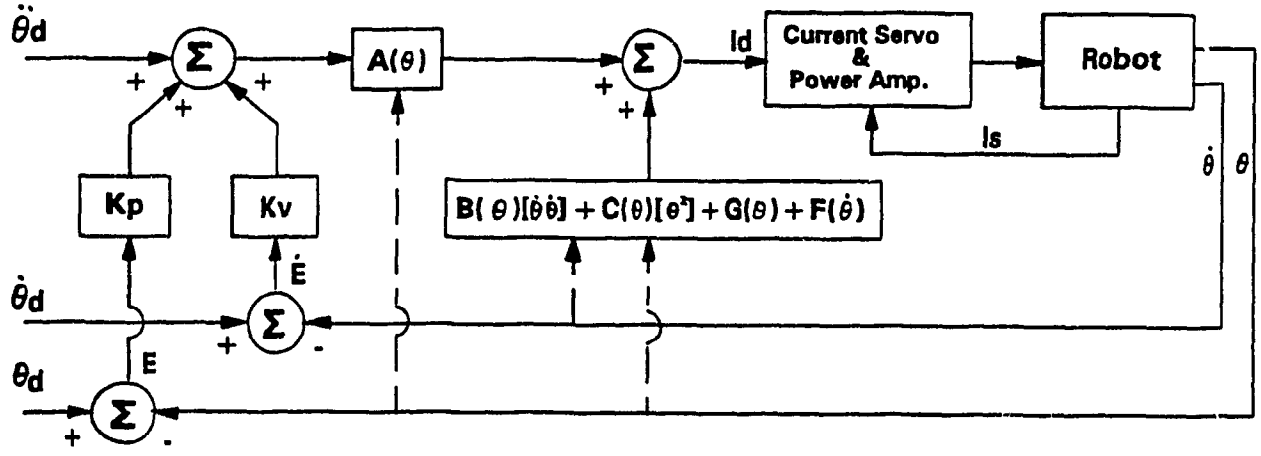


Fig. 4.3 Block Diagram of the Overall Feedbck Control

4.2.5 Serial/Parallel Interface and Power Supplies

4.2.5.1 Serial/Parallel Interface

The serial/parallel interface, the counters and other related circuitry constitute the

innermost control loop in the model-based robot controller architecture. As mentioned before, to overcome this problem, a general purpose interface between a serial digital device and a 8-bit parallel digital device has been earlier designed by Simon & Gilles. For complete details reader is referred to [71] and only some essential features of the interface circuitry are discussed here. An Inmos IMS C011 link adaptor [47] forms the heart of the S/P interface by converting bi-directional serial link data into parallel data streams. The IMSC011 is operated in peripheral interface mode (mode 1). In this mode it provides eight bit parallel input interface, eight bit parallel output interface and full handshaking signals for both input and output. On one side (serial side) the IMSC011 is connected to the CTA through one of the transputer links. On the other side (parallel side) the IMSC011 is interfaced to the peripheral devices through programmable logic arrays (PLAs).

In the current implementation, S/P interface has been used to interface six LM629 motion controllers functions as digital counters of shaft encoders, six AD7828 A/D converters, six DA800C D/A converters [72], one for each joint of the robot, and a number of general purpose digital input/output latches to the transputer. Their addresses are given in Appendix C. The reader should notice the address ALL_COUNTER (0x0f) which is common to all counters for the estimate of the real positions. Thus a START command written on this address will start motion on all the joint servos (programmed with a valid trajectory) simultaneously. The A/D converter is mainly used for reading potentiometer voltages to determine the absolute position of the arm at any time. It can also be used for reading other analog signal from the environment, like the desired speed

of the manipulator etc. The digital latches can be used for varied purposes. The digital inputs can be used to receive triggers from the environment to start synchronisation with a particular arm. Digital outputs may be used to set/reset other subordinate devices.

A typical write operation for a peripheral device from the CTA end looks like the following.

```
chan_out_byte(WR_CMD(DAT)_CMD | (channel << 4), LinkxOutput);
chan_out_byte(command or data, LinkxOutput);
chan_out_byte(RESET, LinkxOutput);
```

The first chan_out byte is for the interfacing PLA. Here channel carries the address of the peripheral device being accessed and Linkx (where x is between 0 and 3) denotes the transputer link being used for communication between JSC and S/P interface. It tells the PLA that wether the byte following is a command or data and for which slave device. As a result, the PLA selects the proper port (command port or data port) of the desired device. The second chan_out command contains the actual byte to be transmitted to the slave device. Here the PLA acts like a transparent buffer. The third chan_out command is used to reset the PLA. The PLA deselects the slave device and returns to a known initial state. A similar scheme is used for reading data from the peripheral device. Several general purpose shell functions like **wr_cmd, rd_dat_byte, rd_port, wr_port** etc. which are used for communication between the CTA and the peripheral device have been developed. These can be called by any program to perform peripheral I/O.

4.2.5.2 Servo Power Amplifier

Although the author attempted to develop such a servo power amplifier that the current through DC motor could be controllable based on the power amplifier coming along with the VALL controller, it turned out to be difficult because of the too many phase shift of the signal, which caused the system unstable sometimes. Advanced Motion Controls' servo amplifiers are used in the implementation [49]. It is designed to drive brush type DC motors at a high switching frequency. It is fully protected against over-voltage, over-current, over-heating and short-circuits across motor, ground and power leads. Several operating modes are provided for different applications. The current mode allows the current through the motor to be controllable. Loop gain, current limit, input gain and offset can be adjusted using 15-turn potentiometers. The limit of the rate current is set as 4 amperes, while the peak current is set as 6 amperes according to [33]. The calibration of the servo amplifiers are included in Appendix D.

4.3 Hardware Architecture

The real-time computational demands of the model-based control as outlined before, is very high. As a result to use an independent processor for each layer of the controller hierarchy discussed above.

The resulting transputer network configuration consisting of six processors is shown in Figure 4.4 where the symbolic name given to each processor is indicated outside the block. In the same figure, the task mapped on the specific processor is labelled inside the block. It is important to note that although the model-based controller

Table 4.1 Link Interconnections

Link Interconnection			Communicating Tasks
No.	Processor From	Processor To	End Point Tasks
1	ROOT #0	PC	FILTER - SERVER
2	ROOT #1	T212	GMS Control Link
3	ROOT #2	IOSERV #1	SERVER - MUX
4	IOSERV #2	MASTER #1	CPP - SERVER (OPERATOR interaction)
5	IOSERV #0	TC560 #3	SERVER - KIN560 (Debugging Link)
6	IOSERV #3	CT560 #3	SERVER - DYN560 (Debugging Link)
7	MASTER #2	CT560 #3	CPP - DYN560
8	MASTER #0	TC560 #0	CPP - DYN560
9	MASTER #3	CUBSPL #3	CPP - INTPOL
10	CT560 #0	E0 (Edge Connector)	DYN560 - S/P INTERFACE
11	CUBSPL #0	IOSERV #X	INTPOL - SERVER (Debugging Link)

architecture demands only four transputers, two extra processors are required to provide debugging support during the development cycle. These are processors ROOT and IOSERV in Figure 4.4. The reason for this has been discussed in Section 4.2.1. Apart from task CPP, which interacts with the operator, such an arrangement provides two free debugging links. These can be connected to two different tasks allowing the use of Parallel C run-time library (which includes I/O functions like printf etc.) in these tasks This provides the essential debugging support required in the development phase. However, in the end product the only task requiring the I/O support for operator

interaction, i.e. CPP, can be mapped on ROOT and resulting network shall require only five processors. The link connections required for establishing transputer network shown in Figure 4.4 are given in Table 4.1. The table also gives the tasks communicating through the various links. The link connections 1,2,3,4 and 7 are default connections on the motherboard TMB08 (refer section 4.2.1). Connections 5,6,8,9 and 10 are established by programming IMS C004 Crossbar Switch using the ncs utility described in the same section. Appendix G gives the configuration file and the ncs connect file for the implemented network.

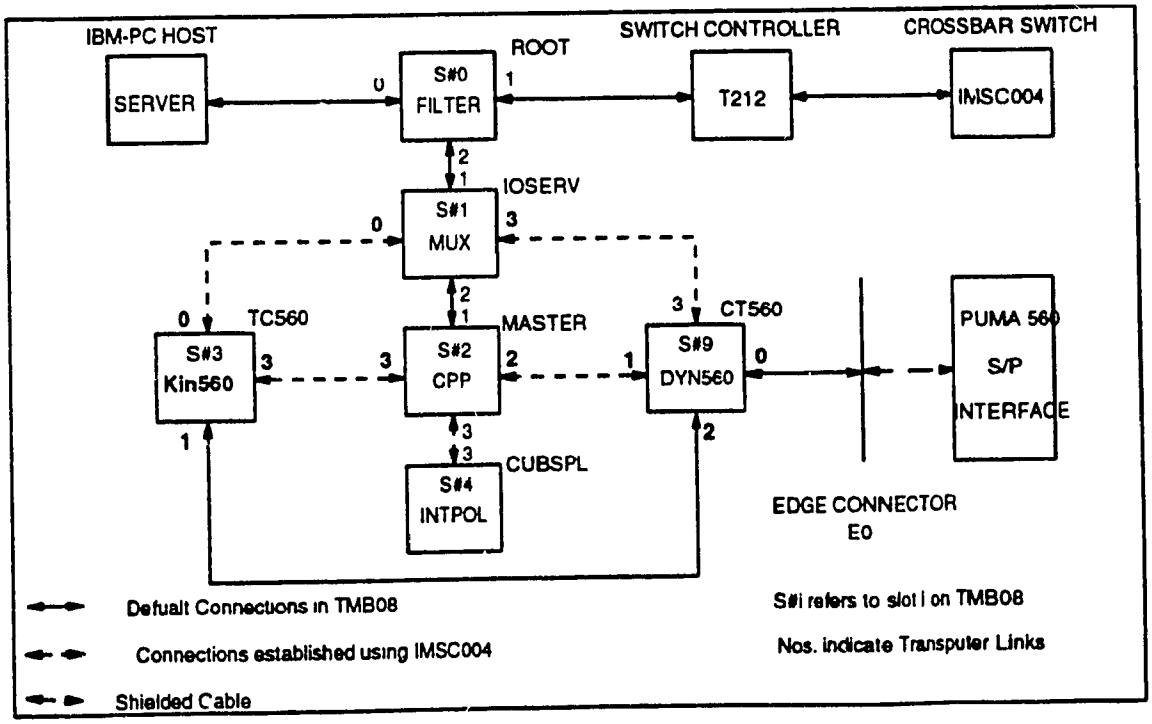


Fig. 4.4 Transputer Network Configuration

4.4 Software Architecture

The software of the controller consists of different tasks which are mapped onto specific processors and execute in parallel (refer to Appendix B). The mapping and the link configuration between various tasks has been described in the last section. This section will describe the function of each task in detail.

4.4.1 Cartesian Path Planning

This task runs on the processor master, which interacts with the operator and functions as a task planner for tasks such as calibration, independent joint motion, a specified motion in cartesian space. After initialisation, on operator command communicates with task TC every T_c (25 ms) seconds. During this interval, it determines the path to be pursued by the tool manipulator. And this information i.e. the next cartesian set point, to the task TC.

4.4.2 Trajectory Conversion

Trajectory Conversion has been discussed in section 4.2.3. Once the desired point on the preplanned path is received from the task CPP, it find the inverse kinematics solution, and check if the desired joint position or positions are achievable. And according to the results, it will pass the desired joint position to the CT control Algorithm, or, it will pass an error massage to the next layer of the controller to shut down the power, and to the OPERATOR for further debugging.

4.4.3 Computed Torque Control Algorithm

CT Algorithm functions to generate the control action for the robot according to the CT control law described in Equation (4.5). This task runs on processor CTA and communicating with the TC and the S/P interface. At each sample instant, it receives the next desired joint position, velocity and acceleration vector from the TC, read the estimated current joint position and velocity vector from the shaft encoder via the S/P interface, calculate the sin's and cos's and all the number crunching operations to generate the control action according to the control law described in Equation (4.5). Though only TC communicates with CT Algorithm in the current implementation, the latter uses the alt_wait_vet command to determine the communicating port for latter expanding.

The tasks **FILTER** and **MUX** are vendor supplied and have been discussed in Section 4.2.1.

4.5 Summary

To summarise this chapter, the design, architecture and the implementation of the model-based robot controller on a network of six transputers has been described in a considerable detail. The essential issues involved in the trajectory conversion from the cartesian space in terms of the end effector of the robot to the corresponding trajectory in joint space, robot dynamical modelling, the computed torque control, the software system are high-lighted.

Chapter 5

Experimental Results

5.1 Introduction

This chapter presents the real-time experiment results of the developed controller, using a Puma 560 robot as the test machine. Extensive experiment is performed at very different speeds, with the maximum speed ranging from 0.15 rad/s to 1 rad/s in the joint space, 0.2 m/s to a 0.92 m/s in the cartesian space of the robot. And the maximum speed used in the test almost reaches the speed limits of the electro-mechanical design of the robot arm system [2]. Performance comparison is made between the traditional PID control and the computed torque control scheme both in the joint space and cartesian space of the robot at normal speed. Extensive experiment is also done to investigate the effect of the sampling rate and the speed of the trajectory on the tracking performance of the developed controller.

The traditional PID control was implemented successfully using the transputer technology []. In order to compare the performance of CT control and that of PID control, PID control scheme is also re-implemented. The control of the robot using the two different control schemes is accomplished by a simple programable switch.

Section 5.2 describes the joint space experiment results, while the desired trajectory is specified for each joint of the robot, including a description of the desired trajectory, performance comparison with the PID controller, investigation of the effect

of the sampling rate on the tracking performance, and the effect of the speed of the trajectory on the tracking performance. Section 5.3 presents the experimental results in cartesian space, while the desired trajectory is specified in terms of the end effector of the robot in the cartesian space.

5.2 Joint Space Experiments

5.2.1 Desired Trajectory

A fifth order polynomial in the joint space, described by Equation (5.1) was used to generate the desired trajectory for each joint of the robot.

$$\begin{aligned} \theta_{d\,i} &= at^3 + bt^4 + ct^5 + (\theta_0)_i \qquad i=1,2,3 \\ 3\dot{\theta}_{d\,i} &= 3at^2 + 4bt^3 + 5ct^4 \\ \ddot{\theta}_{d\,i} &= 6at + 12bt^2 + 20ct^3 \end{aligned} \tag{5.1}$$

where a, b, and c are constant determined by the whole displacement of the trajectory and the time required to complete the trajectory,

$$(\theta_0)_1 = 0.0, \qquad (\theta_0)_2 = -\frac{2}{3}\pi, \qquad (\theta_0)_3 = \frac{1}{3}\pi$$

$$(\theta_0)_4 = -\frac{1}{6}\pi, \qquad (\theta_0)_5 = -\frac{1}{6}\pi, \qquad (\theta_0)_6 = -\frac{1}{6}\pi$$

During the tests actual position and velocity are estimated from the readings of the shaft encoders during each sampling period to determine the peak tracking position and velocity errors through the entire trajectory, as well as the root mean square (RMS) values of the errors of both position and velocity of the operation.

5.2.2 Tracking Performance

Figures 5.1 - 5.12 show the trajectory position error and velocity error of joints 1 to 6 of the robot respectively by using the computed torque control. It is a sample of the experimental results. Each joint is commanded to track 1.05 radians (60 degrees) during 5 seconds independently along the desired trajectory described by Equation (5.1) above.

Table 5.1 Performance of a Typical Experiment in Joint Space

Joint No	Position error in rad. $x10^{-2}$ (Magnitude)				Velocity error in rad/s $x10^{-2}$ (Magnitude)			
	Maximum		RMS		Maximum		RMS	
	PID	CT	PID	CT	PID	CT	PID	CT
1	1.12	0.31	0.71	0.17	2.04	2.83	0.88	0.87
2	1.91	0.56	1.2	0.26	2.85	3.24	1.15	1.41
3	1.59	0.24	0.98	0.11	2.23	2.39	0.78	0.98
4	1.13	0.59	0.70	0.27	3.96	1.62	0.99	0.75
5	1.32	0.29	0.72	0.17	2.62	2.52	0.79	1.38
6	1.12	0.38	0.70	0.20	2.32	6.1	0.90	2.2

The maximum position, velocity error and their corresponding RMS values for

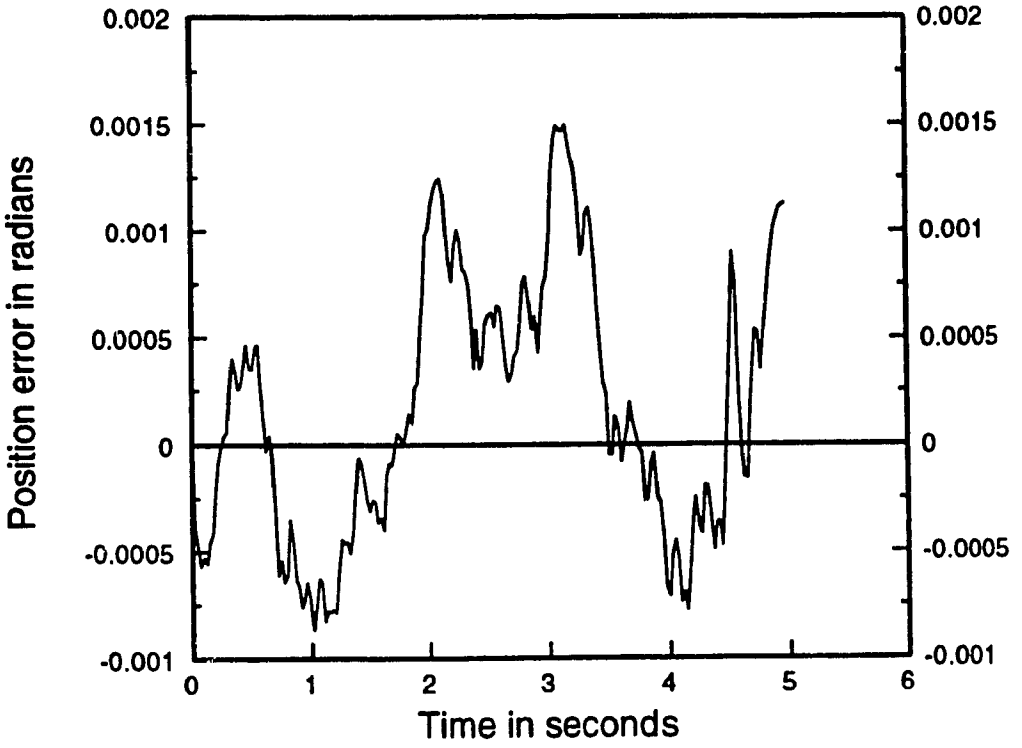


Fig. 5.1 Joint 1 Position Error (Joint Space Exp.)

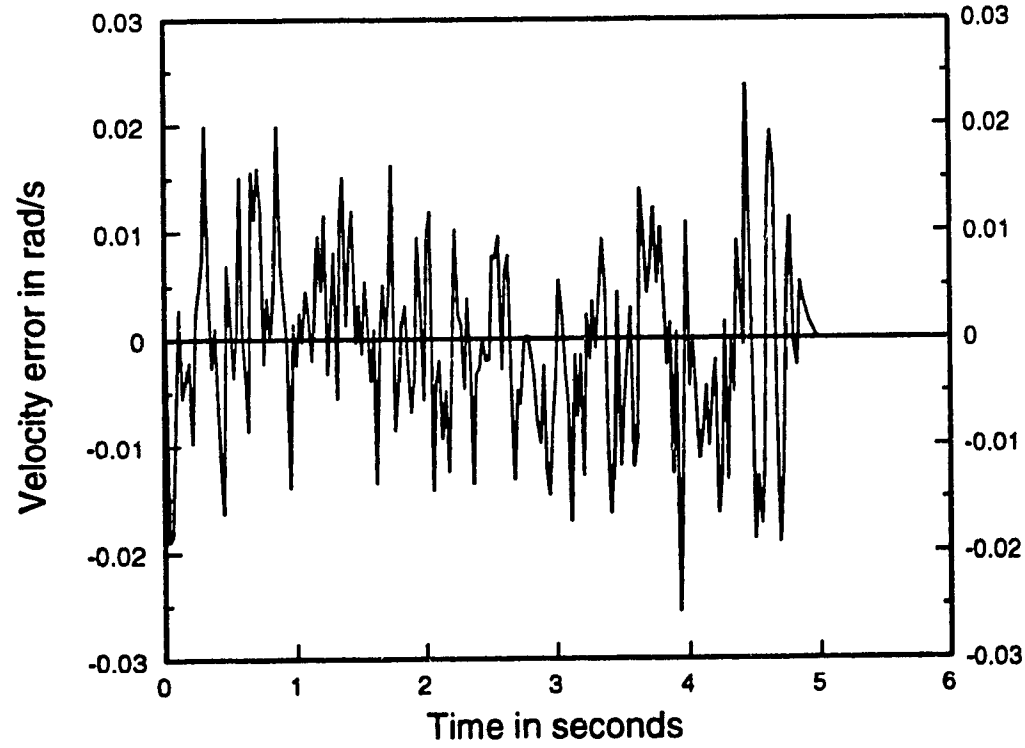


Fig. 5.2 Joint 1 Velocity Error (Joint Space Exp.)

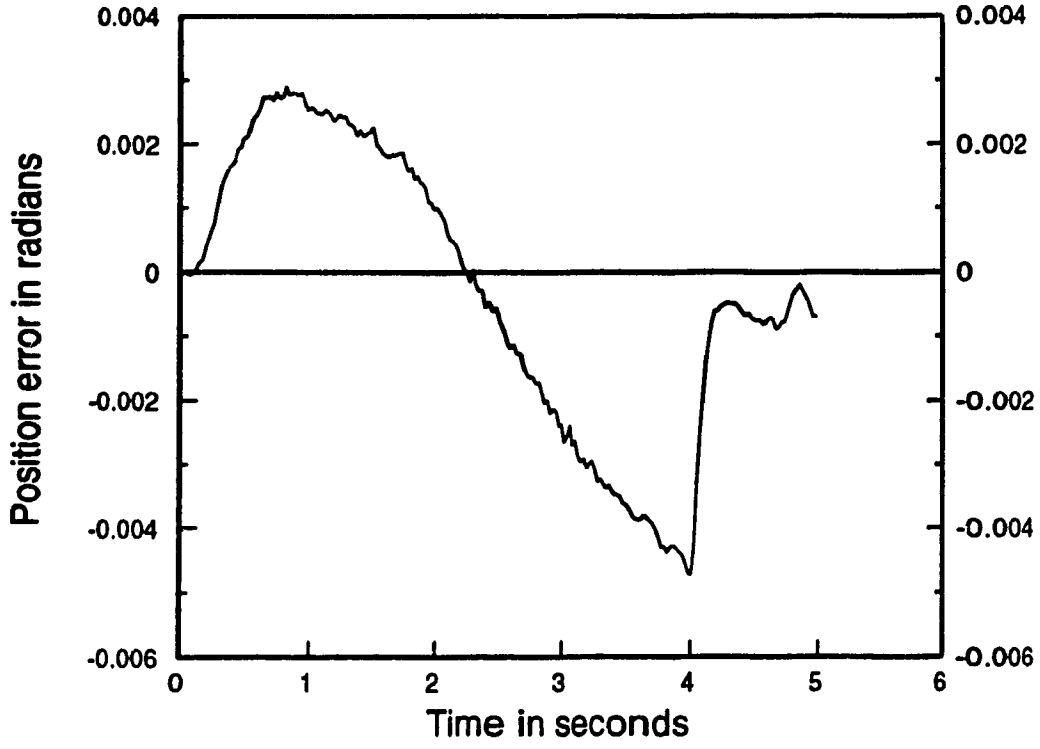


Fig. 5.3 Joint 2 Position Error (Joint Space Exp.)

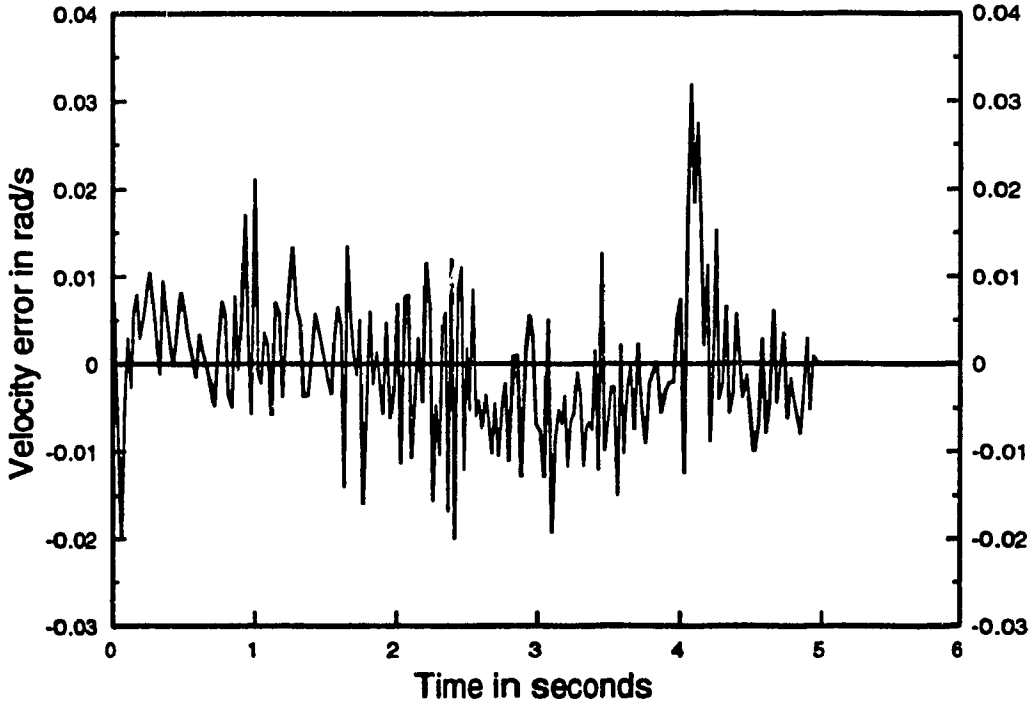


Fig. 5.4 Joint 2 Velocity Error (Joint Space Exp.)

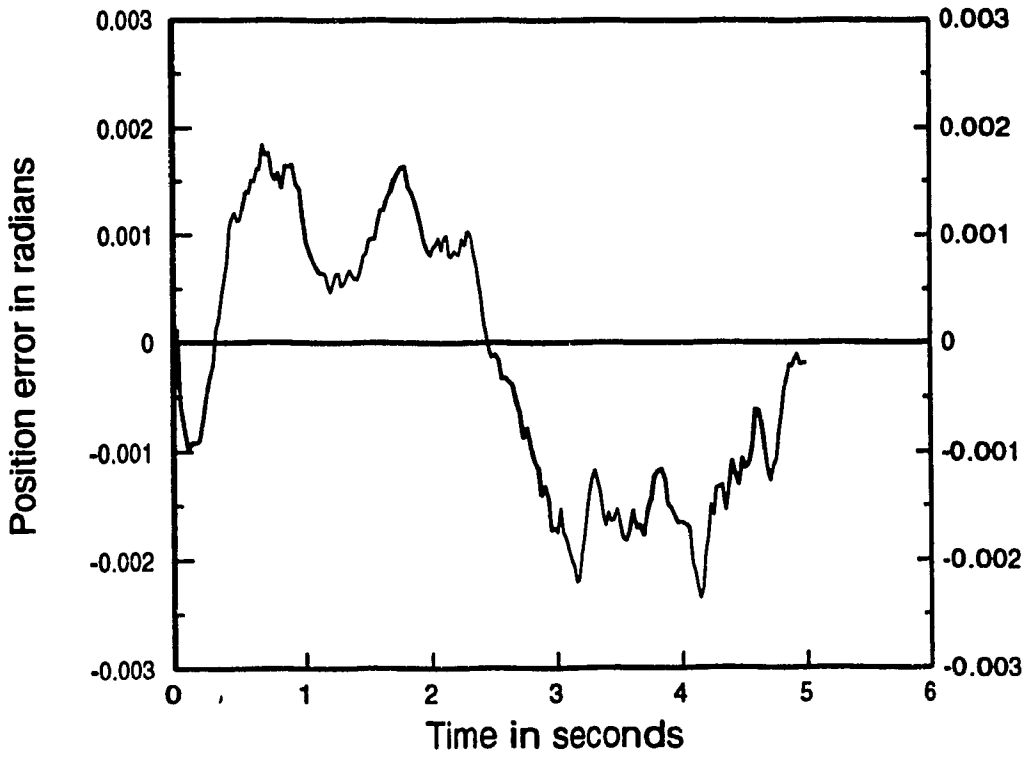


Fig. 5.5 Joint 3 Position Error (Joint Space Exp.)

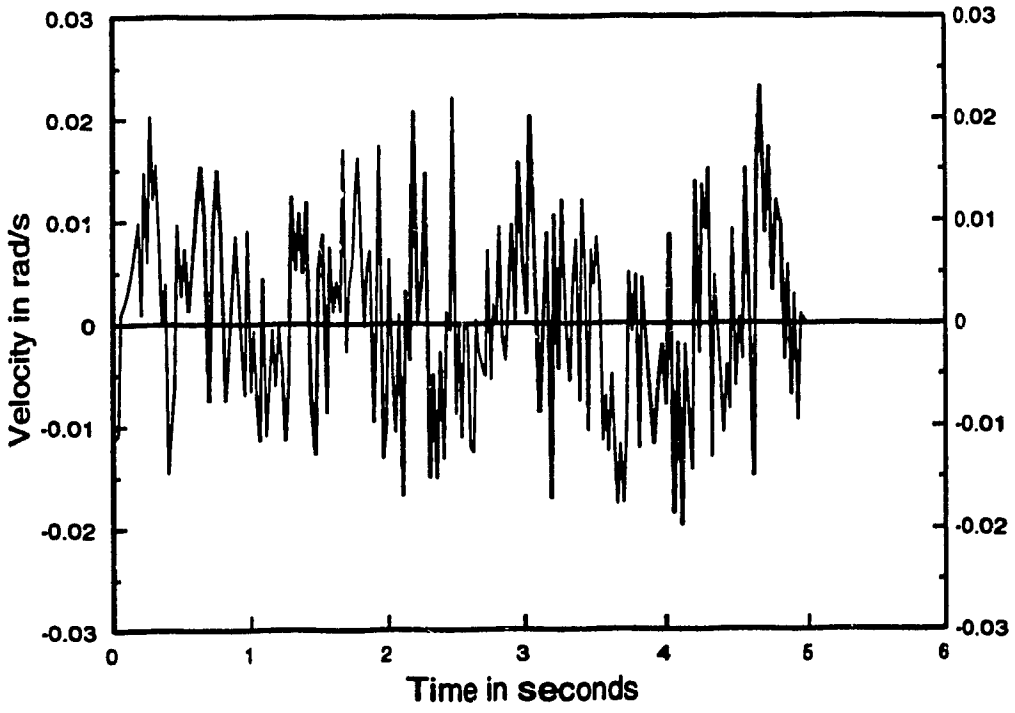


Fig. 5.6 Joint 3 Velocity Error (Joint Space Exp.)

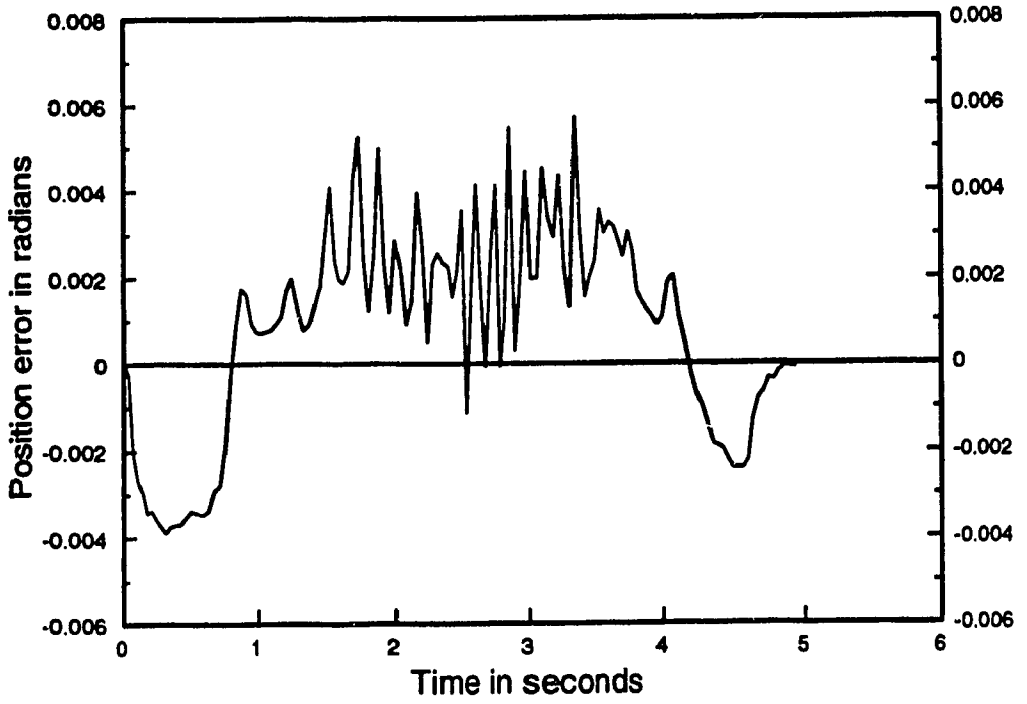


Fig. 5.7 Joint 4 Position Error (Joint Space Exp.)

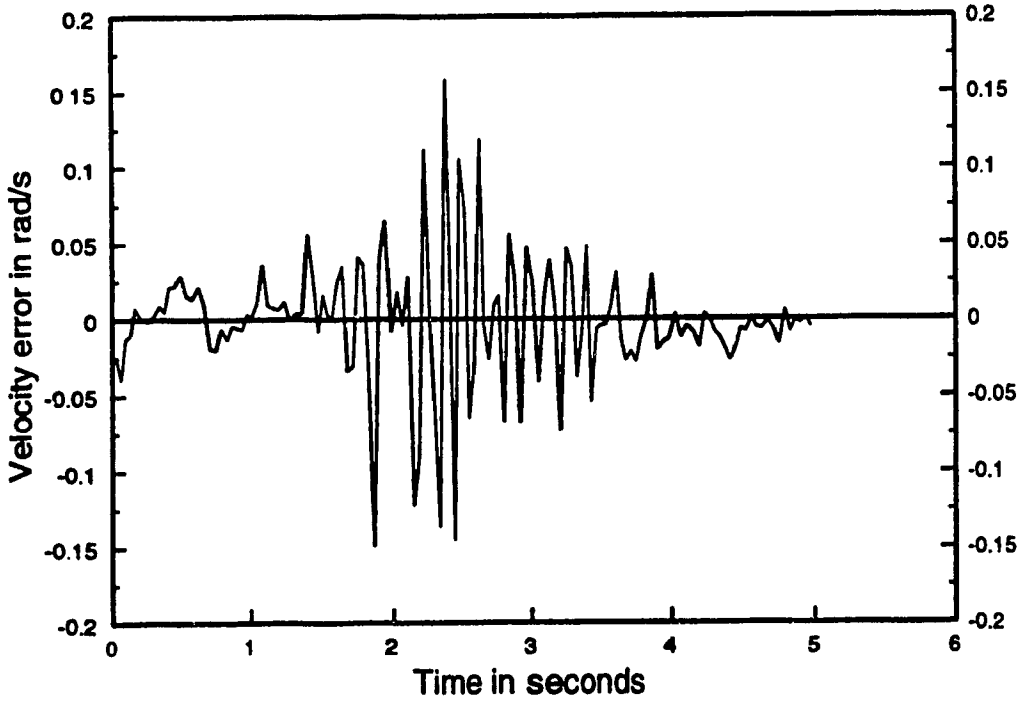


Fig. 5.8 Joint 4 Velocity Error (Joint Space Exp.)

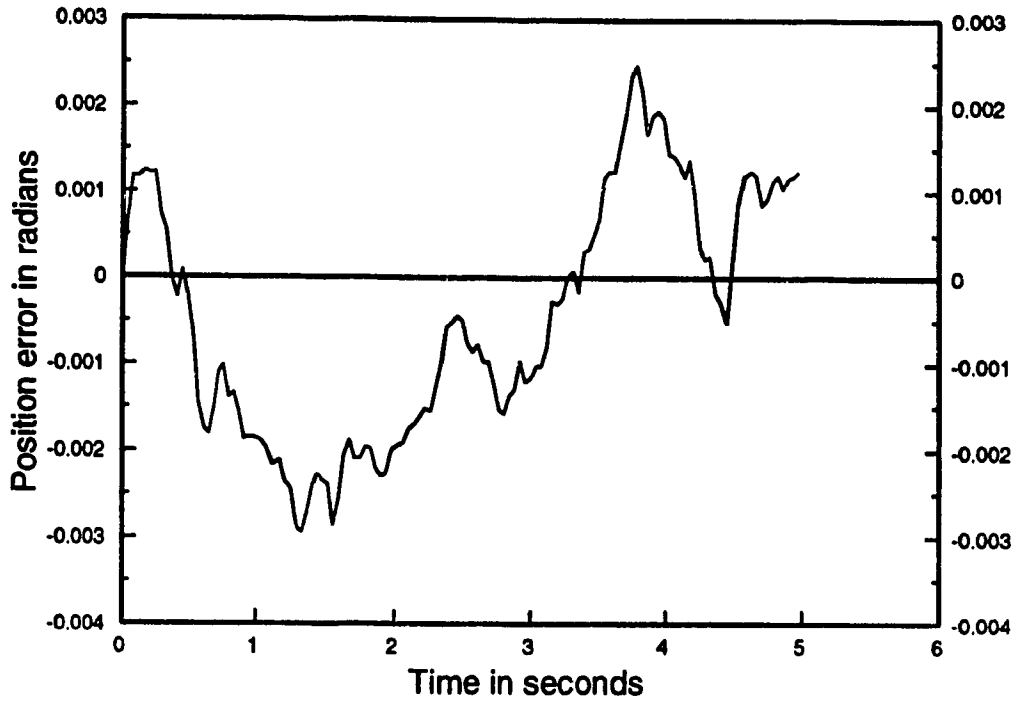


Fig. 5.9 Joint 5 Position Error (Joint Space Exp.)

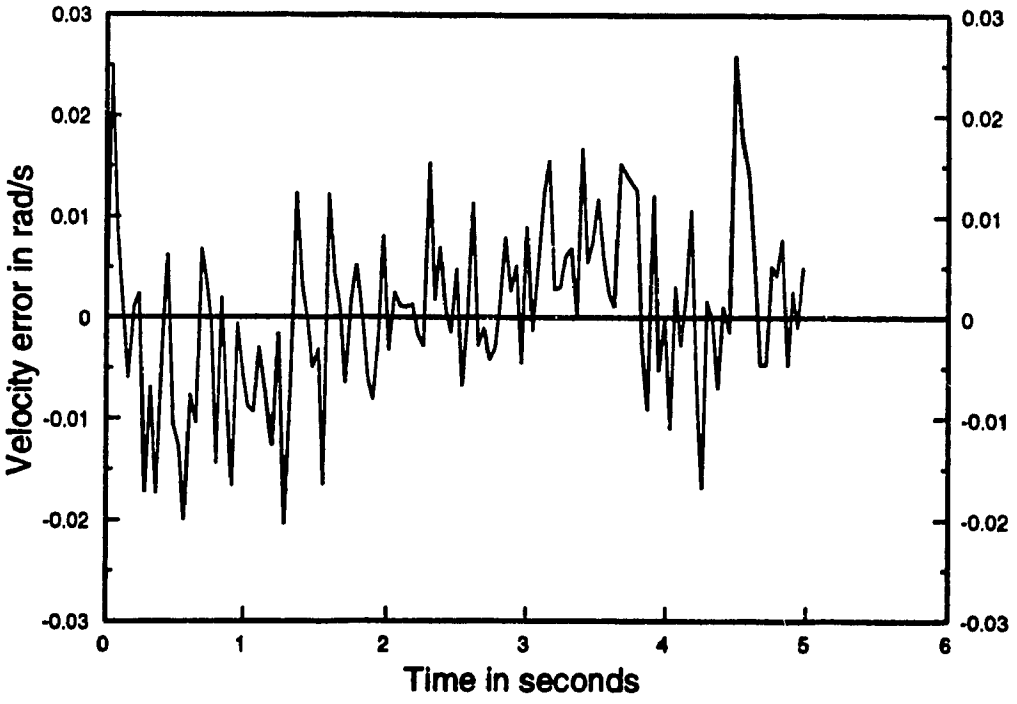


Fig. 5.10 Joint 5 Velocity Error (Joint Space Exp.)

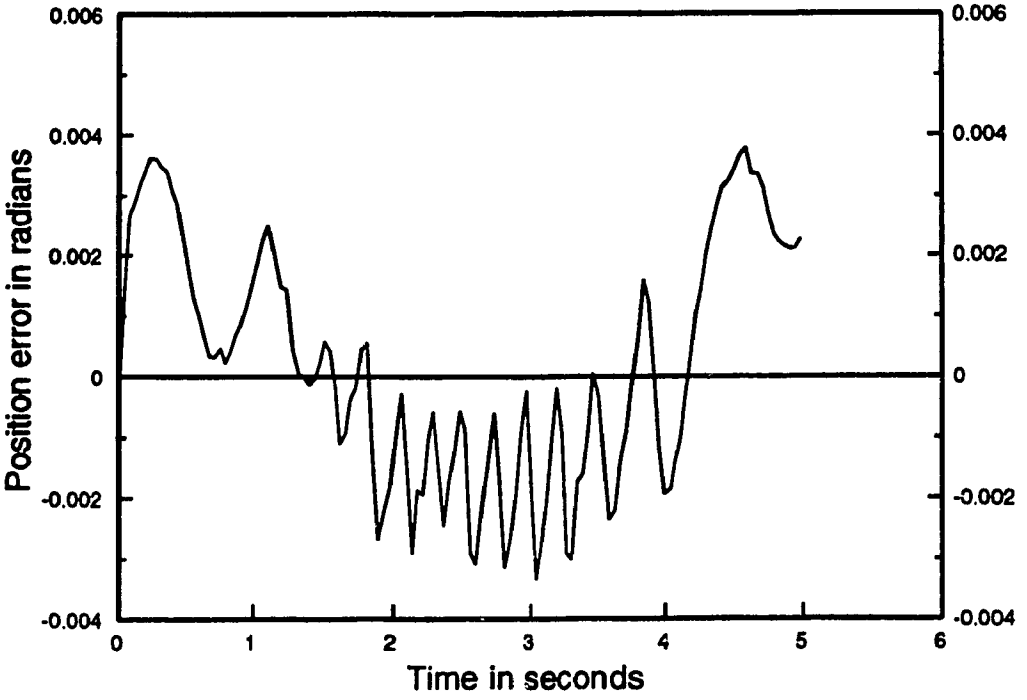


Fig. 5.11 Joint 6 Position Error (Joint Space Exp.)

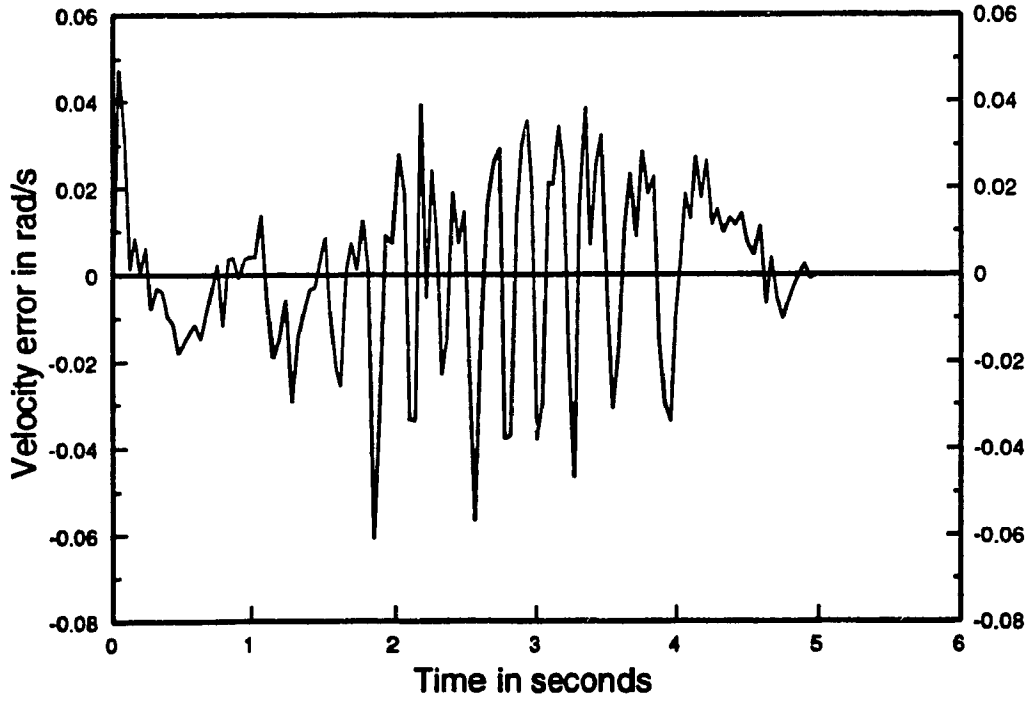


Fig. 5.12 Joint 6 Velocity Error(Joint Space Exp.)

the above experiment is given in Table 5.1. The experiment of the traditional PID control scheme using the same desired trajectory is done. And the results are also listed in the table. The sampling period of the CT control is 1 ms, and that of the PID control 256 μs.

It can be seen that the computed torque control outperforms the traditional PID control in terms of the position errors. The position error of the CT control is about 30-50% of the PID control both in terms of the maximum value and the RMS value of the position error.

5.2.3 Effect of the Sampling Period on the Performance

As mentioned in Chapter 3, the issue of effect of the sampling rate of the computed torque control on the trajectory tracking performance of the robot has been the subject of much discussion, and in particular, experimentally studied in [16] by using a simple mechanical load as the test rig. That the higher tracking performance of the robot is obtained by using a higher sampling rate was observed. It also showed that there was an upper cut-off point of the sampling rate, above which the improvement of the tracking performance was only marginal, and further improvement of the performance was only obtained by the accuracy improvement of the dynamical model of the robot used in the control. Thus a series of experiments have been designed and conducted to address this issue. Each experiment employs a different sampling rate. The natural frequency of the system is tuned to get the best performance at that particular sampling rate without the observation of any vibration of the robot arm.

Figure 5.13 shows the maximum position error, maximum velocity error, and the RMS value as a function of the sampling period. It is built from 10 set of experiments. It can be seen that the higher trajectory tracking performance is obtained by using higher sampling rate. However, when the sampling rate reached 500 Hz, hardly any performance improvement can be seen with the increase of the sampling rate. When the sampling rate is some value between 200 and 500 Hz, significant improvement of the performance can be seen by increasing the sampling rate. The tracking performance of the controller degraded very quickly when the sampling rate slowed down, starting from some value around 100 Hz.

5.2.4 Effect of the Speeds of Trajectory on Performance

Varying the time required to complete the desired trajectory while maintaining the amplitude of displacement of the desired trajectory constant changes the instantaneous value of the desired velocity and acceleration. The effect of the speeds of the trajectory on the tracking performance is shown in Figure 5.14, both for the traditional PID and the computed torque control, which covers the range from a slow of 0.15 rads/sec to an exceedingly high of 1 rad/sec. It can be seen that the computed torque controller is quite robust, showing little degradation of performance with varying speeds. However, the performance of the traditional PID control deteriorate very fast with the increase of the speeds of the trajectory.

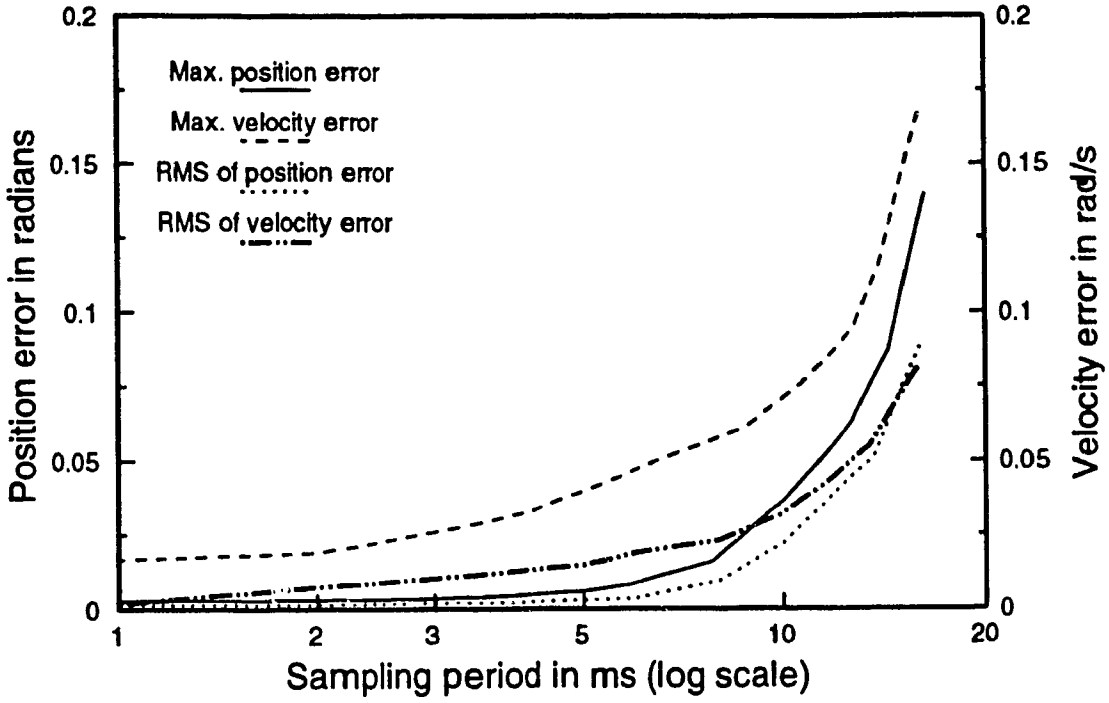


Fig 5.13 Effect of Sampling Period on Performance

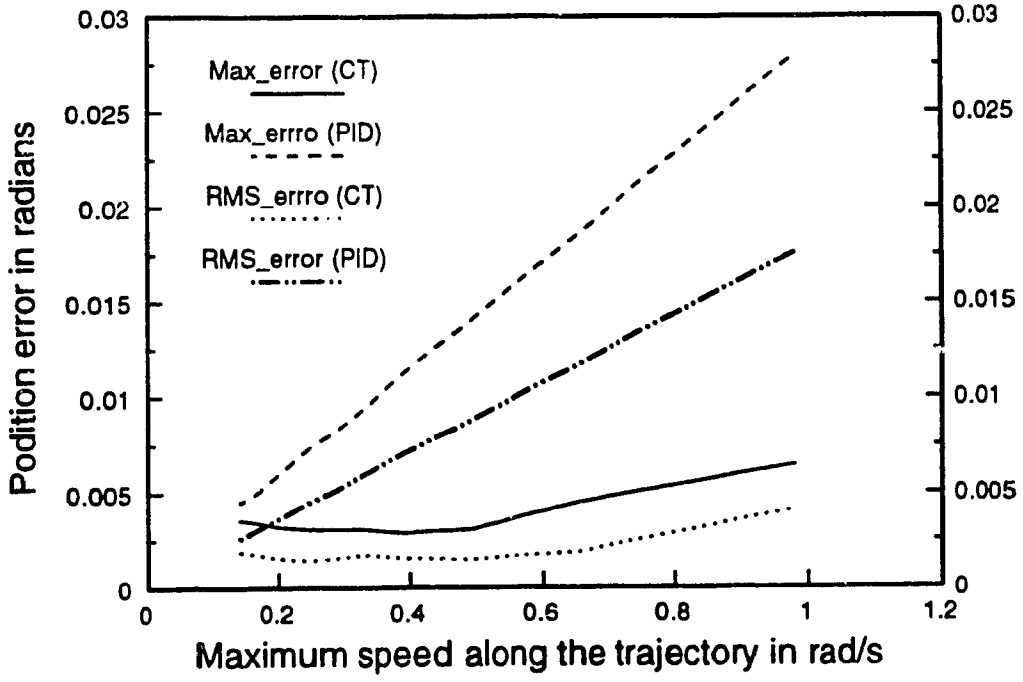


Fig. 5.14 Effect of the Speed on Performance

5.3 Cartesian Space Experiments

5.3.1 Desired Trajectory

The robot is commanded to move with gravity from a starting point (0, 430, 530) to (-400, 600, 300) with the tool orientation right up, and goes back against gravity, following a fifth order polynomial profile. Each joint of the robot has a displacement of varying between 40 - 50 degrees, except for joint 4 of which the desired joint displacement is zero. As the gravity of the arm and the friction of the driving system dominate the dynamics of the Puma 560 Arm, the desired trajectory designed is able to show the effect of the dynamics compensation of the CT control.

5.3.2 Real-time Performance at a Typical Speed

The maximum linear velocity along the desired trajectory is 0.2 m/s which is about the half speed of the maximum linear velocity specified by VALL II controller. Figures 5.15 - 5.26 show the position error and velocity error profiles of joints 1 to 6 of the robot respectively.

The maximum position, velocity error and their corresponding RMS values by using both the computed torque control and the traditional PID control scheme for the sample experiment are given in Table 5.2. Compared the performance of the CT control with traditional PID control, same observation can be made as in the joint space experiments described in section 5.2.2. This is very reasonable as the robot is controlled in joint space even the task is specified in cartesian space.

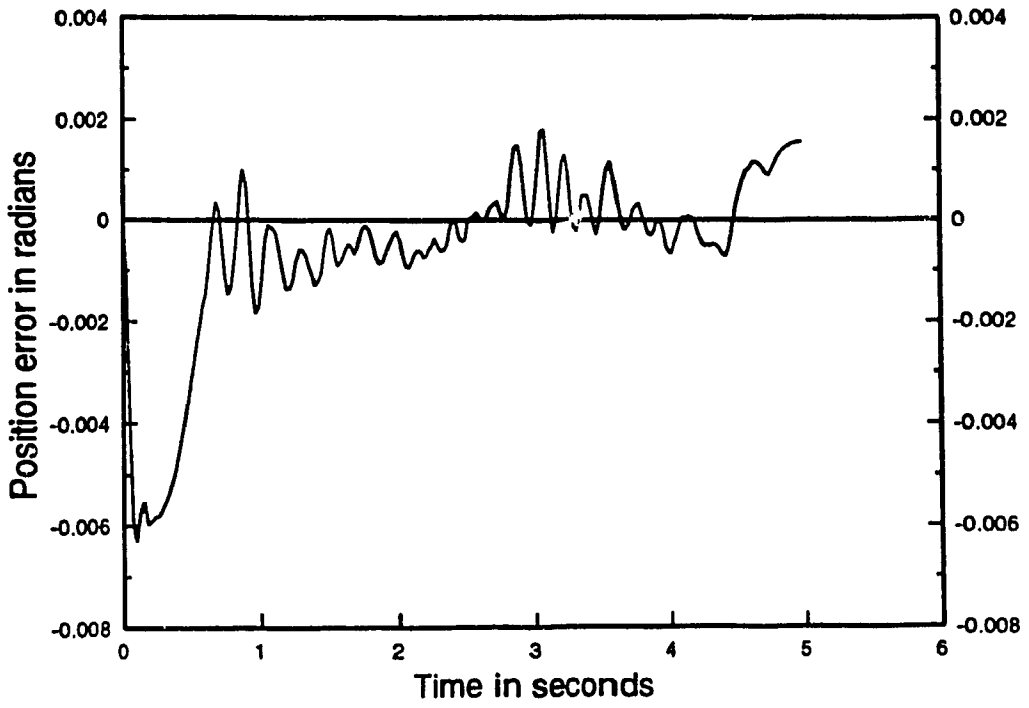


Fig. 5.15 Joint 1 Position Error (Cart. Space Exp.)

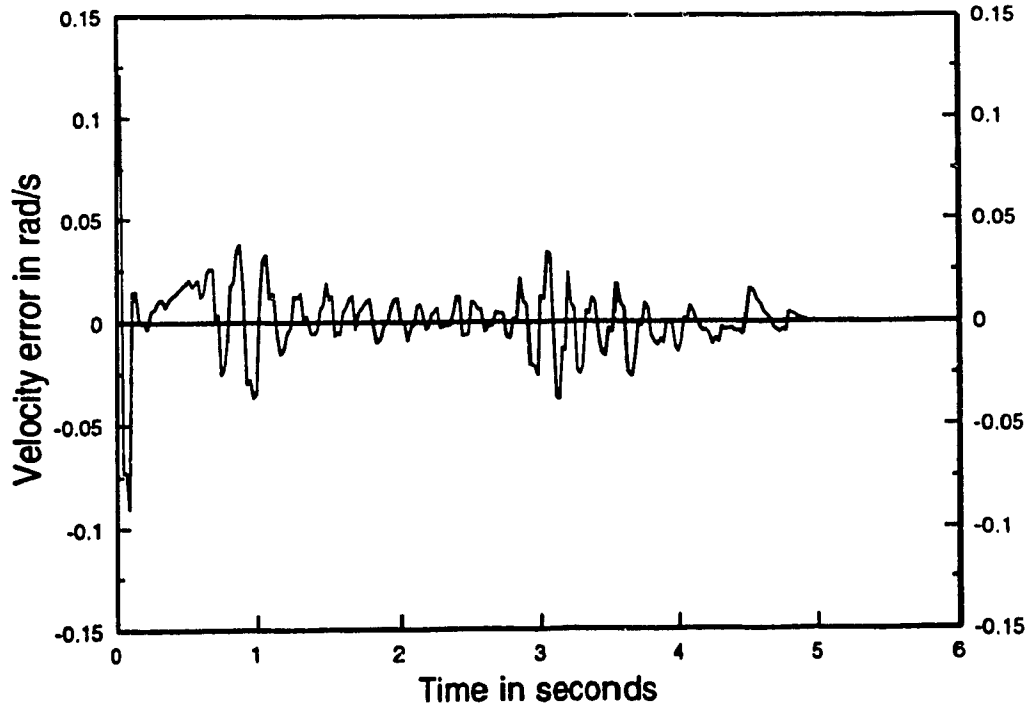


Fig. 5.16 Joint 1 Velocity Error (Cart. Space Exp.)

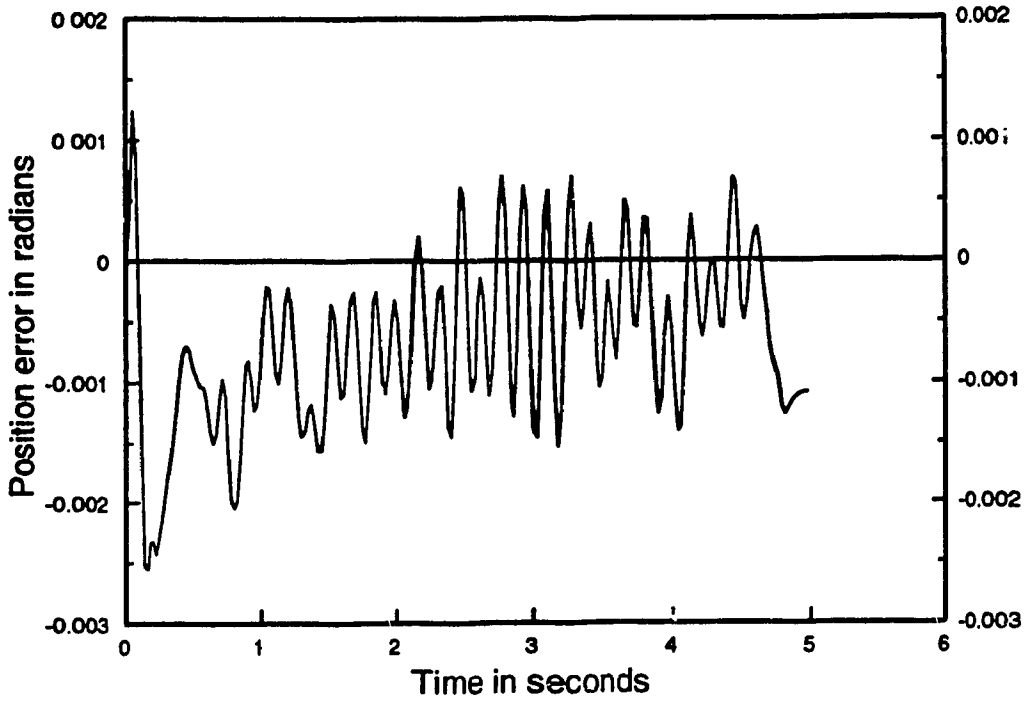


Fig. 5.17 Joint 2 Position Error (Cart. Space Exp.)

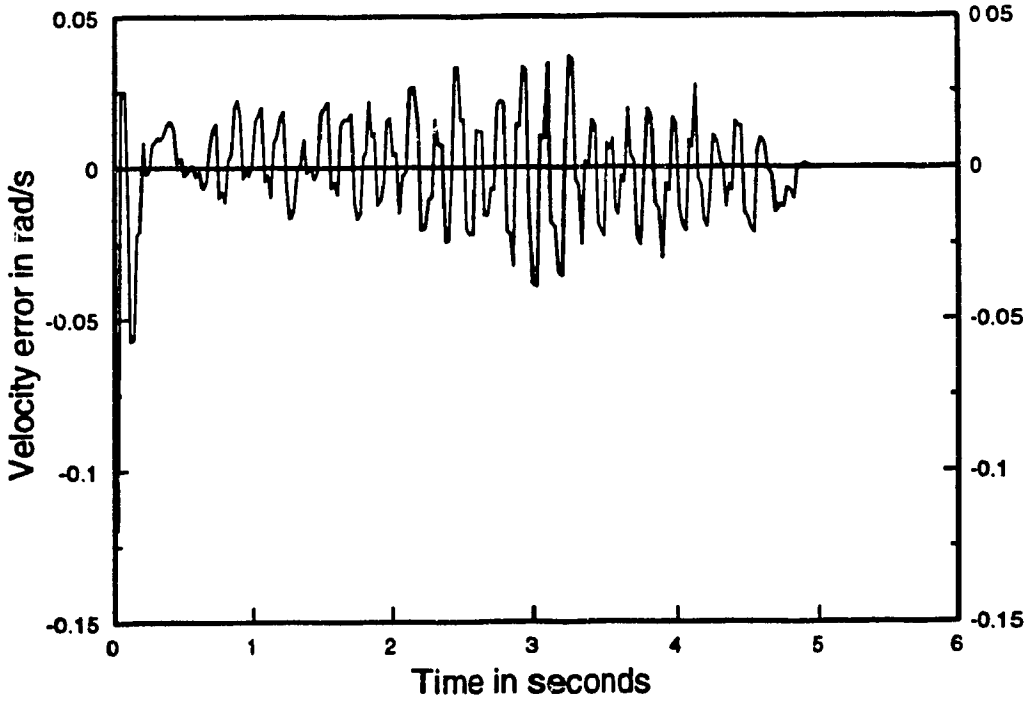


Fig. 5.18 Joint 2 Velocity Error (Cart. Space Exp.)

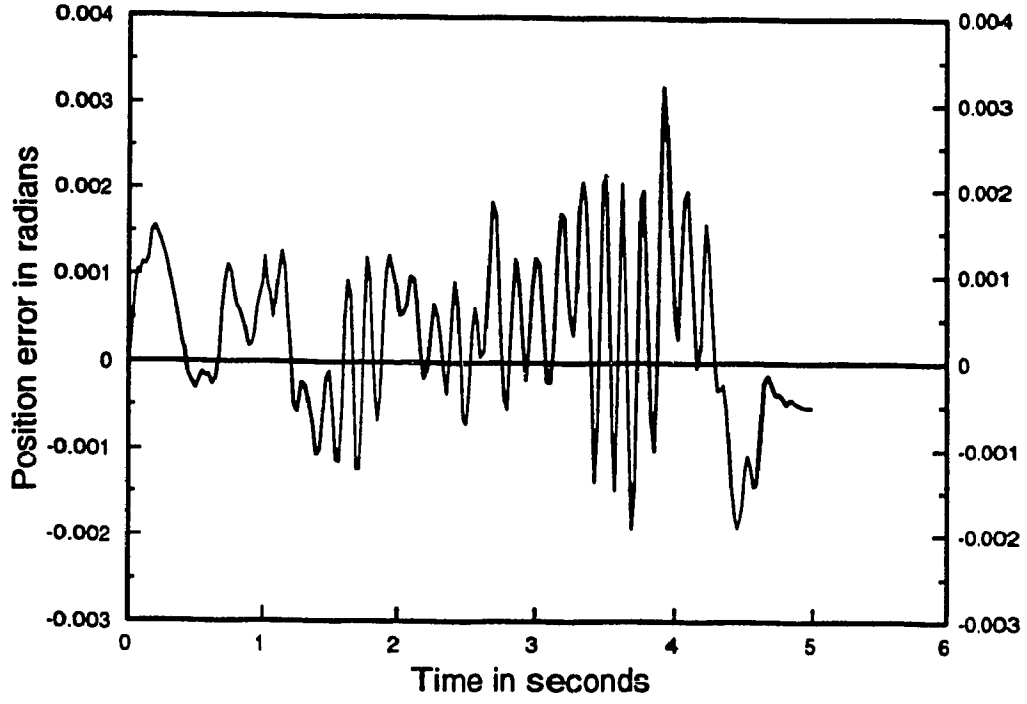


Fig. 5.19 Joint 3 Position Error (Cart. Space Exp.)

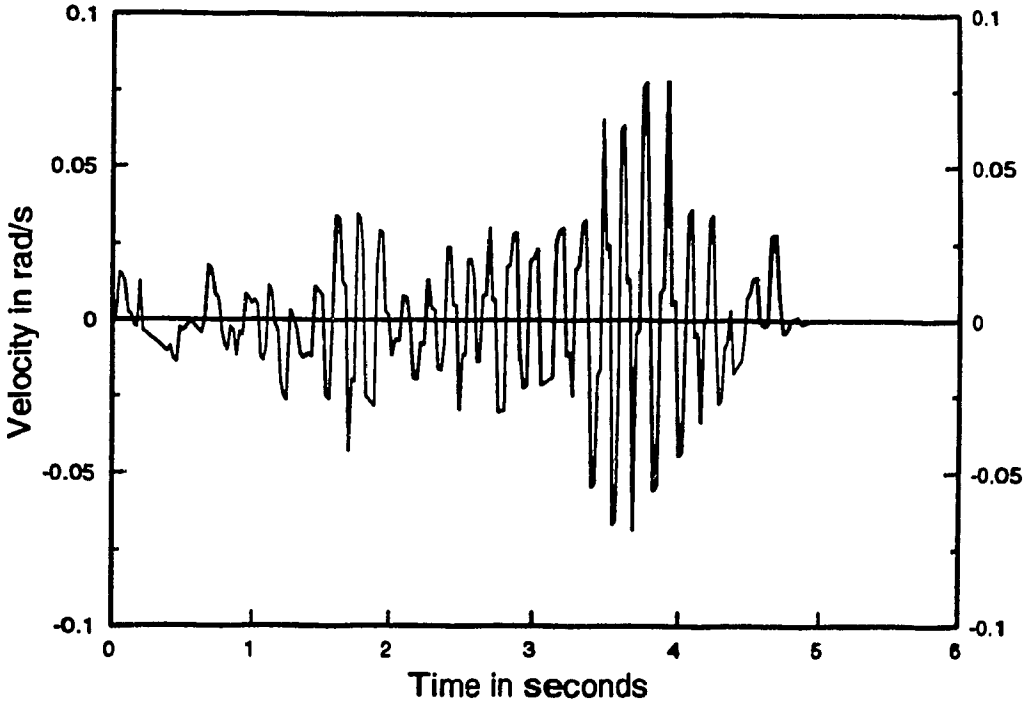


Fig. 5.20 Joint 3 Velocity Error (Cart. Space Exp.)

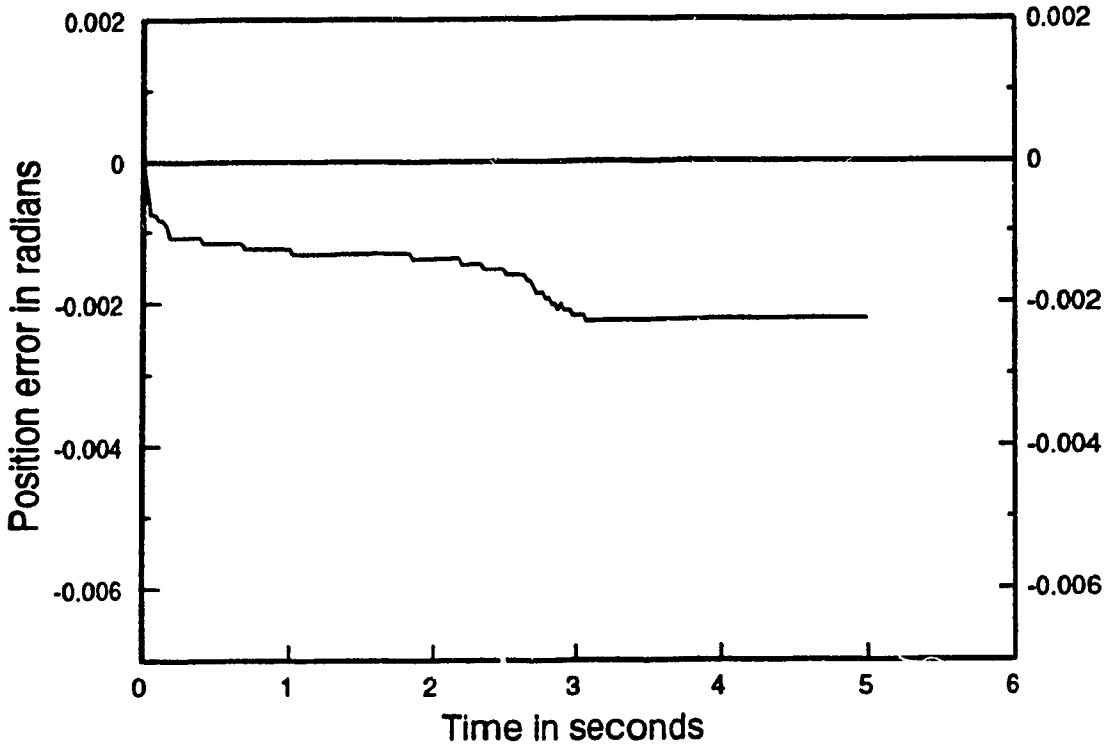


Fig. 5.21 Joint 4 Position Error (Cart. Space Exp.)

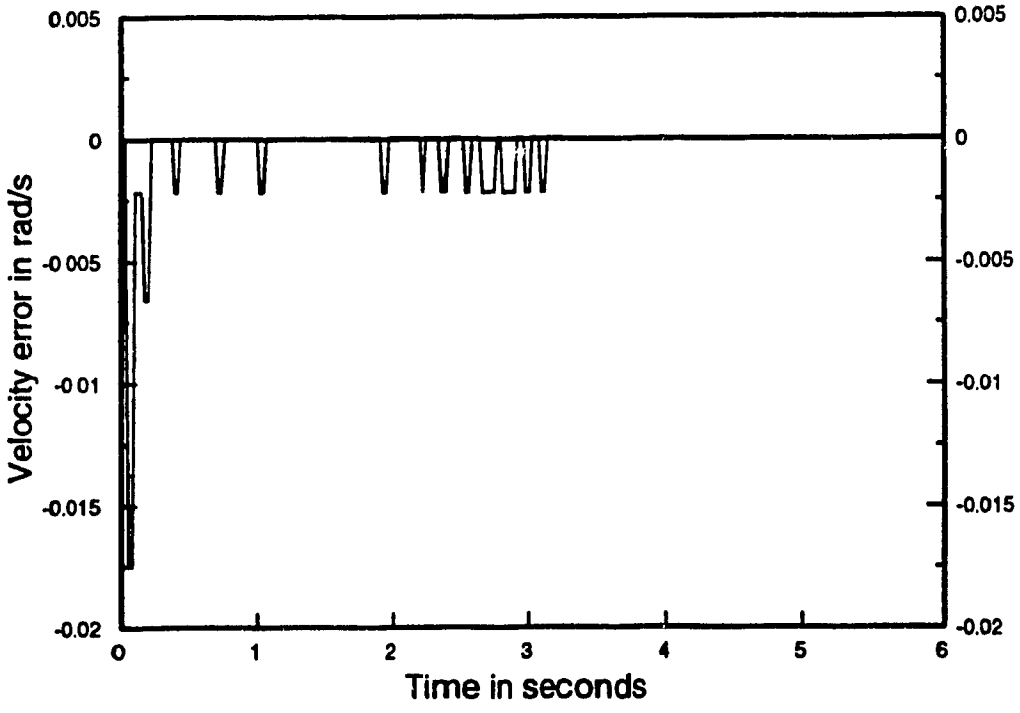


Fig. 5.22 Joint 4 Velocity Error (Cart. Space Exp.)

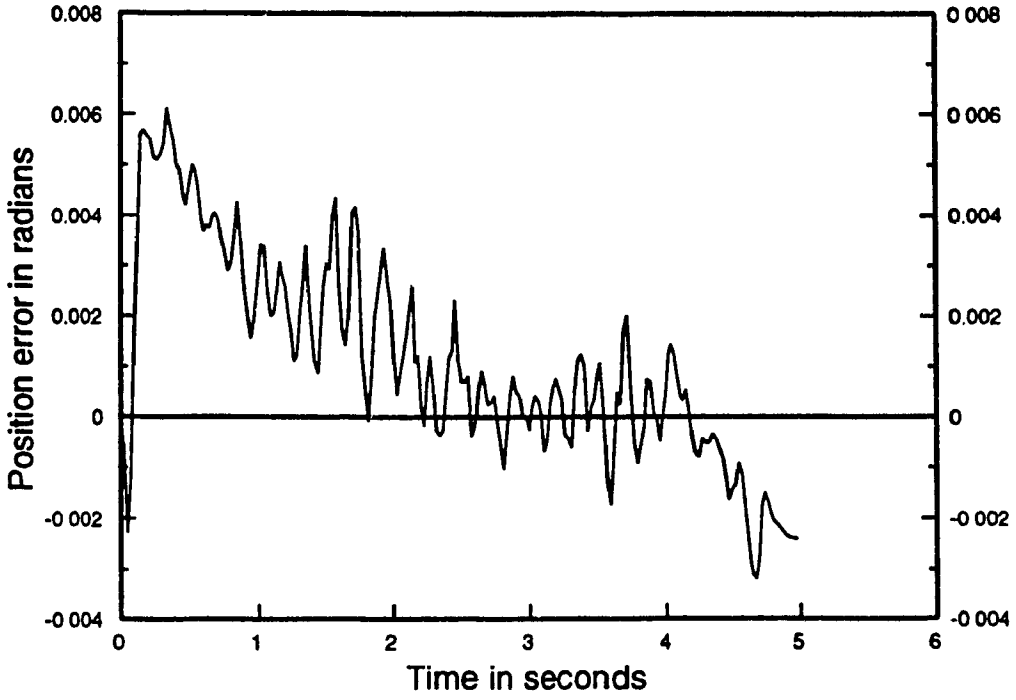


Fig. 5.23 Joint 5 Position Error (Cart. Space Exp.)

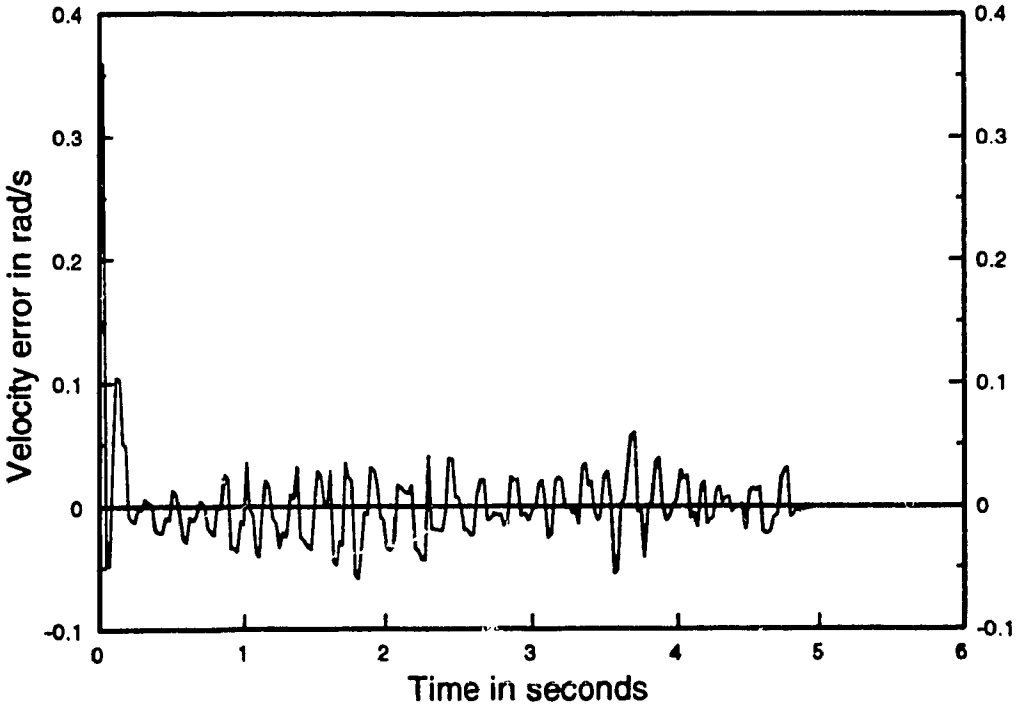


Fig. 5.24 Joint 5 Velocity Error (Cart. Space Exp.)

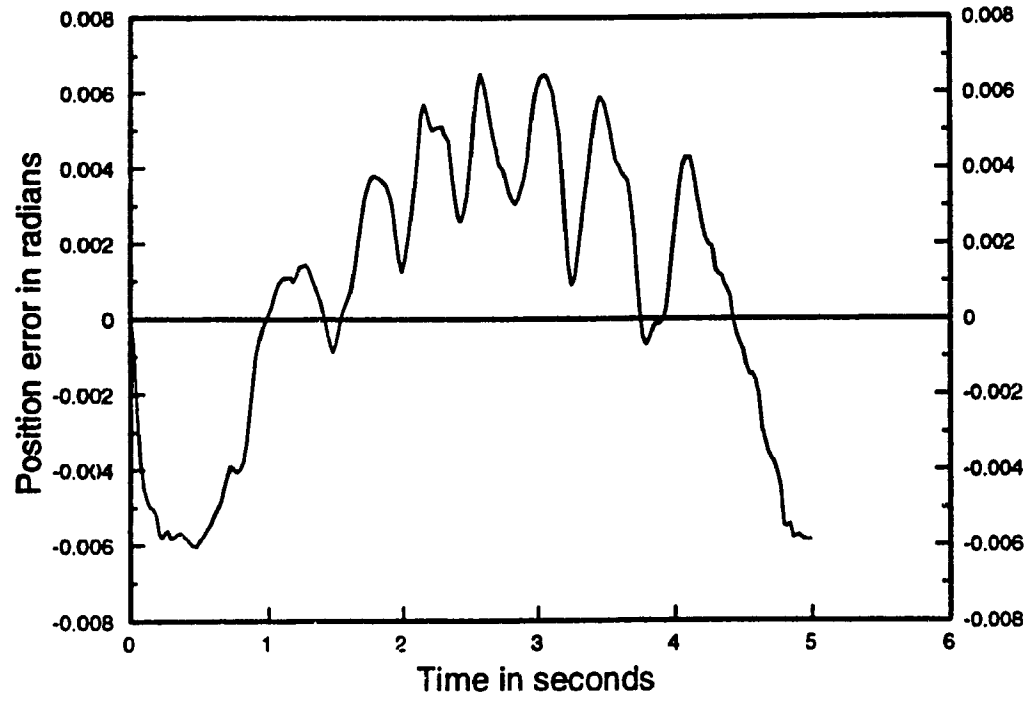


Fig. 5.25 Joint 6 Position Error (Cart. Space Exp.)

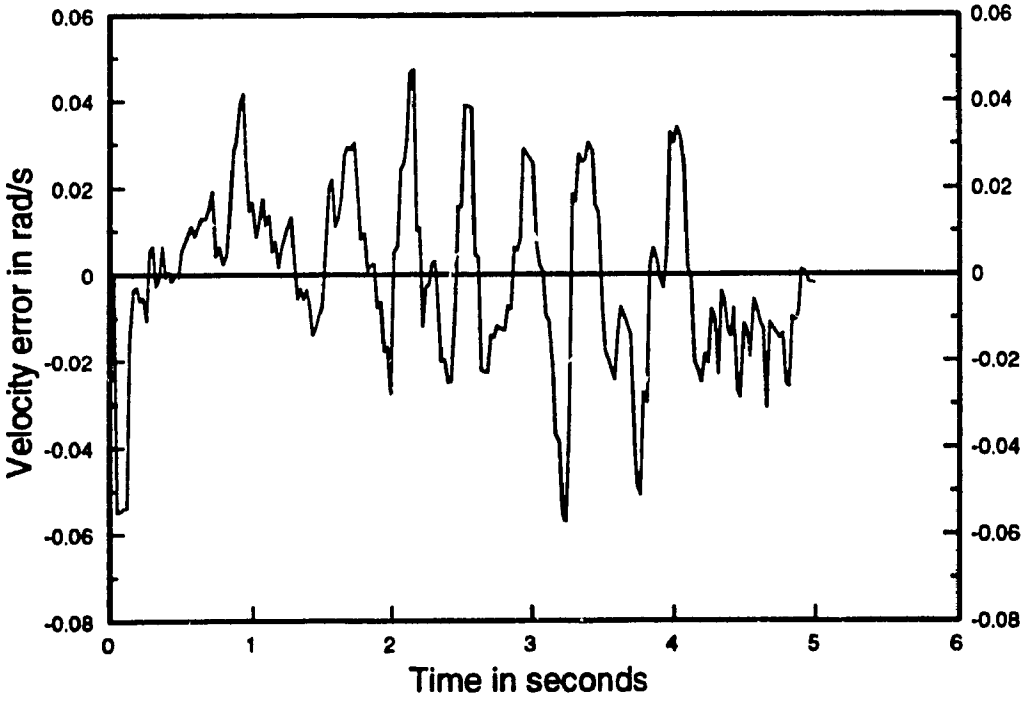


Fig. 5.26 Joint 6 Velocity Error (Cart. Space Exp.)

Table 5.2 Performance of a Typical Experiment in Cartesian Space

Joint No	Position error in rad. $x10^{-2}$ (Magnitude)				Velocity error in rad/s $x10^{2}$ (Magnitude)			
	Maximum		RMS		Maximum		RMS	
	PID	CT	PID	CT	PID	CT	PID	CT
1	1.78	0.63	0.84	0.17	2.19	12.07	0.94	1.98
2	1.96	0.76	1.23	0.41	2.44	11.97	1.03	1.99
3	1.43	0.32	0.93	0.14	2.20	7.97	0.78	2.55
4	0.76	0.23	0.45	0.18	4.10	1.75	1.05	0.20
5	1.84	0.62	0.99	0.24	2 71	3.59	0.83	1.48
6	1.53	0.65	0.82	0.38	2.43	5.72	0.92	2.03

5.3.3 Real-time Performance at High Speed

By reducing the time period to complete the desired trajectory, the robot is operated at a very high linear velocity of 0.936 m/s which almost reaches the maximum linear velocity of the Puma 560 Arm (1 m/s). This figure has never been reported so far. By using the controller provided by the vendor, a maximum linear velocity of 0.46 m/s is achieved. The maximum velocity and acceleration for the first three joints is around 50 degree/sec and 200 degree/s/s respectively, while the maximum velocity and

acceleration for joint 5 and joint 6 is around 30 degree/s and 120 degree/s/s respectively. Joint 4 is commanded to stay still for the desired trajectory.

Even operating the robot at the above mentioned high speed neither obvious vibration of the Arm, nor audible noise was observed. The Maximum position errors, velocity errors, and the corresponding RMS values are listed in Table 5.3, while the performance for PID control of a Puma 560 robot at such a high speed is, of course, not available.

Table 5.3 Performance of CT Control at High Speed in Cartesian Space

Joint No	Position error in rad $x10^{-2}$ (Magnitude)		Velocity error in rad/s $x10^{-2}$ (Magnitude)	
	Maximum	RMS	Maximum	RMS
1	1.27	0.46	12.07	3.12
2	1.39	0.68	11.97	3.01
3	0.94	0.49	9.98	2.82
4	0.24	0.13	1.57	0.23
5	0.82	0.37	5.32	2.30
6	0.72	0.33	6.47	2.18

5.4 Summary

Substantial experimental results have been presented in this chapter, using a Puma 560 Arm as the test rig. Model-based controller, with its dynamic compensation ability, performs much better compared to the traditional PID control when the sampling period of the model-based controller is 1.25 ms, while the later is 256 μs, in terms of the maximum position error and its RMS value along the desired trajectory.

The investigation of the effect of the sampling period of the computed torque control on the trajectory tracking performance has showed that very good performance can be achieved without an extra cost of the computation power requirement at the sampling period between 1 to 2 ms. Performance degradation can be seen with the increase of the sampling period starting from 2 ms. The performance of the controller deteriorate fast with the increase of the sampling period starting from some value around 10 ms for Puma 560 robot.

With the CT controller, Puma 560 robot can reach a maximum linear velocity in terms of the end effector of the robot of 0.92 m/s which is close to the maximum linear velocity of the arm itself. Also there is no obvious vibration of the arm is observed, nor audible noise. With the vendor provided controller (PID), a maximum linear velocity of 0.46 m/s is only achievable.

The computed torque controller is also quite robust, showing little degradation of the performance with vary speeds of the trajectory. But the performance of the traditional PID controller degrades fast with the increase of the speed of the trajectory.

Chapter 6

Conclusions and Recommendations for Future Work

6.1 Conclusions

Most industrial manipulators are highly nonlinear, coupled dynamic systems, low speed, large peak position errors are the direct results of the utilization of traditional PID control acting on each joint of the robot independently. Model-based control, such as computed torque control, uses the dynamics model of the robot, which results in a high operation speed and high tracking performance of the robot. A controller for the realization of a model-based control scheme, such as computed torque control, using the transputer technology has been developed. A sampling rate of 800 Hz is achievable for the real time control of a 6-DOF robot, while the task is specified in the Cartesian Space in terms of the end effector of the robot, which has never been demonstrated before. The experimental results have showed that the developed controller outperforms the traditional independent PID controller by a factor of 2 or 3 at normal operation speed, in terms of the peak position errors along the trajectory, noise and vibration. With the CT controller, a high end effector speed of 0.920 m/s for Puma 560 robot arm is achievable. This speed almost reaches the maximum linear velocity of the end effector of Puma 560 robot arm. It doubles the maximum end effector speed of the same robot controlled by VAL

controller provided by the vendor, and 5 - 6 times faster than that controlled by the controller presented by [6].

The effect of the sampling rate on the tracking performance is extensively investigated in this thesis. The higher the sampling rate, the higher the tracking performance. However, the performance improvement is only marginally accomplished by the increase of the sampling rate when the sampling rate is reached to a particular value which is different from one manipulator to another (1000 Hz for Puma 560 robot). Further improvement of the performance is only achievable by accuracy improvement of the dynamic model of the robot. It is found out that the performance degrades rapidly with the decrease of the sampling rate when the sampling rate is lower than a particular value (around 100 Hz for Puma 560 robot).

The investigation of the effect of the speeds of the trajectory on the tracking performance showed that the developed controller is quite robust. Little degradation of the performance is noticed for the computed torque control with the increase of the operation speed. This is because of the effective dynamics compensation of the model-based control. The performance of the PID control degrades quickly with the increase of the operation speed.

6.2 Recommendations for Future Work

1. Payload: As far as the model-based control is concerned, payload of the robot could be modeled, ether in the form of the mass of the tool, or in the form of the reactive force from the cutting material during the machining, into the dynamic model.

In the former case, the solution is straight forward. The last link dynamic parameters can be modified to consider the attached mass as part of the link (assume that the additional mass is attached to the last link firmly). In the latter case, the pay load can be resolved into the Joint Space described by a joint torque vector (τ_p) as any force vector (F) written with respect to the base frame {0} of the robot may be transformed to joint torque vector by the transpose of the Jacobean (J^t) of the robot.

$$\tau_p = J^T F \tag{6.1}$$

Then the control action described in Equation (4.6) can be modified as

$$\tau^* = \hat{A}(\theta)[\ddot{\theta}^*] + \hat{B}(\theta)[\dot{\theta}\dot{\theta}] + \hat{C}(\theta)[\theta^2] + \hat{g}(\theta) + \hat{\tau}_f + \tau_p \tag{6.2}$$

2. Friction: Friction and the gravity of the robot usually dominate the dynamics of industrial robot using geared driving systems. Compared with gravity terms, friction is much more complicated and difficult to model. This is the main reason that there is not a recognizable friction model for industrial robots until now, which is practical for real time implementation, and which can model the friction phenomenon of the driving systems reasonably. Nevertheless, a high enough sampling rate and a reasonable friction model are essential for the implementation of model-based control successfully.

3. Inertial Parameters: As discussed in section 2.2.5, most industrial robots are driven indirectly. The motor torque is amplified using a gear mechanism. Hence the effect of inertial variations due to errors of the inertial parameters at the motor shaft is

reduced by $1/r^2$ where r is the gear ratio. However, effect of the variation (or accuracy) of inertial parameters on the performance still need to be evaluated by real time experiment to find out the perfomance variation with respect to the errors of inertial parameters.

References

[1] M. Brady, *Robotics Science,* The MIT Press, Cambridge, Massachusetts, London, England 1989.

[2] Unimation, *500 Series Equipment and Programming Manual 398P1*, Unimation Inc., April 1984.

[3] M.B.J. Leahy, and G.N. Saridis, "Compensation of Industrial Manipulator Dynamics", *Part 1, RAL Technical Report, No. 101*, RPI, September 1987.

[4] National Semiconductor Datasheet, *LM628/629 Precision Motion Controller*, Application Note 706, National semiconductor Corporation, March 1989.

[5] M.B.J. Leahy, "Dynamics Based Control of Vertically Articulated Manipulators", *IEEE Int. Conference on Robotics and Automation*, 1988.

[6] S. Kabra, *The Development and Analysis of an Integrated Workcell Controller for Multiple-Robot Synchronisation*, Concordia University, M.A.Sc. Thesis, April 1992.

[7] R. Rajagopalan, R.M.H. Cheng and S. Poon, "Parallel Computation of Inverse Dynamics of Robots Employing Transputers", *Proceedings of ISMM Int. Conf. on Parallel and Distributed Computing Systems*, 1990.

[8] Watanabe et al, "Improvement in the Computing Time of Robot Manipulators Using a Multi-microprocessor", *ASME Journal of Dynamics, Systems, Measurement and Control*, Vol. 108, pp. 190-197, 1986.

[9] J.W. Xiao, and R.M.H. Cheng, "Efficient Parallel Computation of Robot Inverse Dynamics", *Proceedings of the 2nd National Applied Mechanisms and Robotics Conf.*, Cincinnati, 1991.

[10] R. H. Lathrop, "Parallelism in Manipulator Dynamics", *International Journal of Robotics Research*, 1985, Vol. 4, (2), pp. 80-102.

[11] J.J. Craig, *Introduction to Robotics, Mechanics and Control*, Second Edition, Addison-Weslay Publishing Company, 1989.

[12] M.B.J. Leahy, K.P. Valavanis, and G.N. Saridis, "Evaluation of Dynamic Models for PUMA Robot Control", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1989.

[13] J.S. Robert, *Fundamentals of Robotics: Analysis and Control*, Prentice Hall, Englewood Cliffs, New Jersey 07632, 1990.

[14] H.An. Chae, G.A. Christopher, J.M. Hollerbach, "Model-based Control of a Direct Drive Arm, Part 2: Control", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1988.

[15] Y. N. Shimony, *Handbook of Industrial Robotics*, John wiley & Sons, Inc., 1985.

[16] K.S. Narendra and A.M. Annaswamy, *Stable Adaptive Systems,* Prentic Hall, Englewood Cliffs, New Jersey 07632, 1989.

[17] K.K.D. Young, "Controller Design for a Manipulator using Theory of variables Structure Systems", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. 8, No.2, February 1978.

[18] S. Dubowsky, and D.T. DesForges, "The Application of Model-Referenced

Adaptive Control to Robotic Manipulators", *Journal of Dynamic Systems, Measurement, and Control*, Transactions of ASME Sept., 1979.

[19] E.S. Wesley, *Industrial Robots Computer Interfacing and Control*, Prentice Hall, Inc., Englewood Cliffs, New Jersey 07632 1985.

[20] R.P. Paul, *Modelling, Trajectory Calculation, and Servoing of a Computer Controlled Arm*, Technical Report AIM-177, Stanford University Artificial Intelligence Laboratory, 1972.

[21] H.An. Chae, G.A. Christopher, J.M. Hollerbach, "Experimental Determination of the Effect of Feedforward Control on Trajectory Tracking Errors", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1986.

[22] K.K. Pradeep, and T. Kanade, "Real-time Implementation and Evaluation of Model-based Controls on CMU DD Arm 2", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1986.

[23] H.An. Chae, G.A. Christopher, J.M. Hollerbach, "Experimental Evaluation of Feedforward and Computed Torque Control", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1987.

[24] K.K. Paradeep, and T. Kanade, "Experimental Evaluation of the Feedforward Compensation and Computed-torque Control Schemes", *Proceedings of the 1986 ACC. AAAC*, Seattle, WA, June 1986.

[25] O. Egeland, "On the Robustness of the Computed Torque Technique in Manipulator Control", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1986.

[26] C. Samson, "One Approach pour la Synthese et L'analyse de la Commande des

Robotics Manipulators Rigides", *Technical Report*, INRIA, France, May 1987.

[27] K.J. Astrom, and B. Wittenmark, *Information and System Science Series: Computer Controlled Systems: Theory and Design*, Prentice-Hall, 1984.

[28] K.K. Pradeep, "Choosing Sampling Rates for Robot Control", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1987.

[29] M.B.J. Leahy, K.P. Valavanis, and G.N. Saridis, "The Effects of Dynamic Models on Robot Control", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1986.

[30] M.B. J. Leahy, G.N. Saridis, "The Effects of Unmodeled Forces on Robot Control", *Proceedings of the 25th CDC*, Athens, Greece, December 1986.

[31] I. P. W., Sillitoe, S. Szumko, "Simulation Modelling of a Puma 560 Arm for Control", *Proceedings of ISA Conf. on Industrial Automation*, June 1-3, 1992, Montreal, Canada.

[32] J.M. Hollerbach, "A Recursive Lagrangian Formulation of Manupulator Dynamics and Comparative Study of Dynamics Formulation Complexity", *IEE Tans. on Systems, Men and Cybernastics*, Vol. SMC-10, pp. 730-736, 1980.

[33] M.B. J. Leahy and G.N. Saridis, "Compensation of Industrial Manipulator Dynamics", *The International Journal of Robotics Research*, Vol. 8, No.4, August 1989.

[34] K.G. Shin, and N.D. McKay, "Minimum-time Trajectory Planning for Industrial Robots with General Torque Constraints", *Proceedings of the Int. Conf. on Robotics and Automation*, pp412-7, 1986.

[35] K. Hashimoto, K. Ohashi, and H. Kimura, "An Implementation of a Parallel Algorithm for Real-time Model-based Control on a Network of Microprocessors", *The International Journal of Robotics Research*, Vol. 9. No. 6, December 1990.

[36] T.J. Tarn, S. Ganguly, A.K. Ramadorai, G.T. Marth, and A.K. Bejczy, "Experimental Evaluation of the Nonlinear Feedback Robot Controller", *Proceedings of Int. Conf. on Robotics and Automation*, 1991.

[37] M. Uebel, I. Minis, and K. Cleary, "Improved Computed Torque Control for Industrial Robots", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1992.

[38] J.P. Karlen, J.M. Thompson, H.I. Vold, J.D. Farrell, P.H. Eismann, "A Dual-arm Dexterous Manipulator System with Anthropomorphic Kinematics", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1990.

[39] R.W. Daniel, and P.M. Sharkey, "Transputer Control of a PUMA 560 Robot via the Virtual Bus", *IEE Proceedings*, Vol. 137, Pt. D, No. 4, July 1990.

[40] P.M. Sharkey, R.W. Daniel, and P. Elosegui, "Transputer Based Real Time Robot Control", *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 1990.

[41] K.K. Pradeep, "Some Experimental Results on Model-based Control Schemes", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1988.

[42] E.P. Lawrence, K. Oussama, and J. Hake, "Joint Torque Sensory Feedback in the Control of a PUMA Manipulator", *Proceedings of IEEE Int. Conf. on Robotics and Automation*, 1989.

[43] B. Armstrong, O. Khatib, and J. Burdick, "The Explicit Dynamic Model and

Inertial Parameters of the PUMA 560 Arm", *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, pp 510 - 518, 1986.

[44] P.M. Sharkey, R.W. Daniel, and P. Elosegui, "Transputer Based Real Time Robot Control", *Proceedings of the IEEE Int. Conf. on Robotics and Automation*, 1990.

[45] C.A. Balafoutis, R.V. Patel, and P. Misra, "Efficient Modelling and Computation of Manipulator Dynamics Using Orthogonal Cartesian Tensors", *Int. Journal of Robotics and Automation*, Vol. 4, No. 6, 1988.

[46] S.C.L. Poon, *The Development and Analysis of an Multiprocessor-based Intelligent Robotic Workcell,* M.Eng. Thesis, Department of Mechanical Engineering, Concordia University, 1989.

[47] Inmos Ltd., *The Transputer Data Book*, Second edition, 1989.

[48] Advanced Motion Control, *Advanced Motion Controls PWM Servo Amplifiers*, 3211 Corte Nakoasi Unut 407, Camarillo, California 93012, USA., 1993.

[49] Transtech reference: TMB08MAN0690, *Transtech TMB08 PC Transputer Motherboard User Guide*, Transtech Parallel Systems Corporation, 1990.

[50] R.M.H. Cheng, S.C.L. Poon, R. Rajagopalan, "Interrupt Driven Transputer File Server", *Proceedings of the 4th Conf. of the North American Transputer Users Group*, Ithaca, 1990, pp. 119-130.

[51] 3L Ltd., *Parallel C User Guide Version 2.2.2*, 1991.

[52] K.S. Fu, R.C. Gonzalez, C.S.G. Lee, *Robotics : Control, Sensing, Vision and Intelligence*, McGraw-Hill International Editions, 1987, pp. 12-81, 62-63, 156-159.

[53] S. Elgazzar, "Efficient Kinematic Transformations for the Puma 560 Robot", *IEEE*

Journal of Robotics and Automation, Vol. RA-1, No.3, Sept. 1985, pp. 142-151.

[54] J. Lloyd, V. Hayward, *Kinematics of Common Industrial Robots*, Robotics 4, 1988, 169-191.

[55] A. Bazerghi, A.A. Goldenberg, J. Apkarian, "An Exact Kinematic Model for Puma 600 Manipulator", *IEEE Trans. on Systems, Man and Cybernetics*, Vol. SMC-14, No.3, May/June 1984, pp. 483-487.

[56] D.E. Orin, W.W. Schrader, "Efficient Computation of the Jacobian for Robot Manipulators", *The Int. Journal of Robotics Research*, Vol. 3, No.4, Winter 1984, pp. 66-75.

[57] C.Z.L. Jim, Yang, D., "An Efficient Motion Control Algorithm for Robots with Wrist Singularities", *IEEE Trans. on Robotics and Automation*, Vol. 6, No.1, Feb. 1990, pp. 113-117.

[58] H. Zhang, R.P. Paul, "A Parallel Inverse Kinematic Solution for Robot Manipulators Based on Multiprocessing and Linear Extrapolation", *IEEE Trans. on Robotics and Automation*, Vol. 7, No.5, Oct.1991, pp. 660-669.

[59] A.Y. Zomaya, et. al., "Fast Robot Kinematic Modelling via Transputer Networks", *Parallel Processing and Artificial Intelligence*, Ed. Mike Reeve, John Wiley & Sons, 1989, pp. 193-213.

[60] R. Rajagopalan, *A Computer-aided Methodology for the Calibration of Industrial Robots Employing Optical Metrology*, M.Eng. Thesis, Department of Mechanical Engineering, Concordia University, 1986, pp. 105-108.

[61] J. Lloyd, *Implementation of a Robot Control Development Environment*, M.Eng.

Thesis, Department of Electrical Engineering, McGill University, 1985.

[62] J.Y.S. Luh, M.W. Walker, and R.P.C. Paul, "Resolved-acceleration Control of Mechanical Manipulators", 1980, *IEEE Tans. Auto. Control*, AC-25(3):468-474.

[63] J.Y.S. Luh, "Conventional Controller Design for Industrial Robot--a tutorial", 1983, *IEEE Trans. System Man Cybernetics*. SMC 13(3): 298-316.

[64] T.R. Kane, and S.A. Levinson, "The Use of Kane's Dynamical Equations in Robotics", 1983, *Int. Journal of Robotics Research* 2(3):3-21.

[65] P.K. Khosla, "Some Experimental Results on Model-based Control Schemes", 1988, *Proceedings of IEEE Int. Conf. Robotics and Automation*, pp 1380-1385.

[66] P.K. Khosla, and T. Kanade, "Experimental Evaluations of Nonlinear Feedback and Feedforwad Control Schemes for Manipulators", 1988, *Int. Journal of Robotic Research* 7(1):18-28.

[67] H.An. Chae, G.A. Christopher, J.M. Hollerbach, *Model-based Control of a Robot Manipulator*, 1988, Cambridge, Mass.: MIT Press.

[68] C.A. Balafoutis, P. Misla, and R.V. Patel, "A Cartesian Tensor Approach for Fast Computation of Manipulator Dynamics", 1988, *Proceedings of IEEE Int. Conf. Robotics and Automation*, pp. 1348-1353.

[69] M. Khalil, J.F. Kleinfinger, and M. Gautier, "Computational Burden of the Dynamic Models of Robotics", 1986, *Proceedings of IEEE Int. Conf. Robotics and Automation*, pp. 525-531.

[70] P.K. Khosla, and C.P. Newman, "Computational Requirements of Customized Newtomized Newton-Euler Algorithms", *Journal of Robotics System* 2(3):309-327.

[71] S.C.L. Poon, G. Huard, *Design of a General Purpose Interface Between a Serial Device and 8 Bit Parallel Device*, CIC Report #0028, Centre for Industrial Control, Concordia University, Montreal.

[72] Analog Devices Datasheet, *LC²MOS high speed 8-channel 8-bit ADC 7828*, Data Conversion Products Databook, Analog Devices, 1988, pp. 3-305 to 3-316.

[73] T.J. Tarn, A.K. Bejczy, G.T. Marth, and A.K. Ramadorai, "Performance Comparison of Manipulator Servo Schemes", on control system magzine.+++

[74] X. Cyril, J. Angeles, and A. Misra, "Efficient Inverse Dynamics of General N-Axis Robot Manipulators", *Proceedings of the Ninth Symp. on the Engineering Applications of Mechanics*, London, Ontario, Canada, 1989.

[75] X. He, and A.A. Goldenberg, "An Algorithm for Efficient Computation of Dynamics of Robotic Manipulators", *Journal of Robotic Systems*, Vol. 7, 1990.

[76] C.J. Li, and T.S. Sankar, "Fast Inverse Dynamics Computation in Real-time Robot Control", *Proceedings of the Annual Conf. of IEEE Industrial Electronics Society*, 1990.

[77] G. Huard, *Controller '2' Puma 560*, CIC Report, Concordia University, Montreal,\ Canada, May 11, 1992.

[78] Tanenbaum, S. Andrew, *Operating Systems : Design and Implementation*, Prentice Hall, New Jersey, 1987, pp. 45-88.

[79] C.A.R. Hoare,, *Communicating Sequential Processes*, Communications of the ACM 21, 1978, pp. 666.

[80] Inmos Ltd., *The Transputer Applications Notebook : Systems and Performance*,

First Edition, 1989, pp. 114-130, 92-113.

[81] Inmos Ltd., *OCCAM 2 Reference Manual*, Prentice Hall Int., 1988.

[82] 3L Ltd., *TBUG User Guide Version 1.0.22*, 1990.

[83] G.M. Asher, M. Summer, "Parallelism and the Transputer for Real-time High Performance of AC Induction Motors", *IEE Proceedings*, Vol. 137, No.4, 1990, pp. 179-188.

[84] H.A. Thompson, P.J. Fleming, "Fault-tolerant transputer based controller configurations for gas-turbine engines, *IEE Proceedings*, Vol. 137, No.4, 1990, pp. 253-260.

[85] L.L. Whitcomb, D.E. Koditschek, "Transputers at Work : Real-Time Distributed Robot Control", *Proceedings of the 4th Conf. of the North American Transputer Users Group*, Ithaca, 1990, pp. 107-118.

Appendix A

Model-based Control of a Simple Mechanical Load

A.1 Introduction

Presented in this appendix is the real-time implementation of the computed torque control on a simple mechanical load. The motivation of this work is to verify the feasibility of the control hierarchy structure presented in chapter 3 by real-time experiment on a simple mechanical load which is designed specially to simulate one link of a robot arm, to investigate the effect of the sampling rate on the tracking performance by experiment to provide further guidance for the controller design and implementation described in Chapter 4.

For the above mentioned objectives, an experimental set up is built based on the designed controller hierarchy, the dynamical model of the mechanical set up is derived by the Lagrangian formulation, a fifth order polynomial is used as the desired trajectory, the effect of the sampling rate, the fastness of the trajectory on the tracking performance is studied by extensive experiments.

This appendix is organized as follows: Section A.2 describes the experimental set up, while a network of T800 transputers is used as the platform, as it will be used for the final design and implementation. Presented in section A.3 is the dynamic modelling, and the computed torque control methodology of the motor mechanical set up. Experiment results are described in section A.4. Summary is presented in section A.5.

A.2 Experimental Set Up

As depicted in Figure A.1, the experiment set up is built to simulate the link and the driving system of a geared industrial robot, which consists of a dc motor with a rate voltage of 24 volts and maximum current of 5 Amperes, coupled with the gear box with a gear ratio of 24.4. And disk A and offset weight B are installed to provide the system inertia and gravity load.

The control system is built based on the designed model-based controller for industrial robots using transputer technology, which is composed of the Host computer, the transputer network, the Serial/Parallel (S/P) interface, the power amplifier with a built in current servo which makes the current of the dc motor controllable.

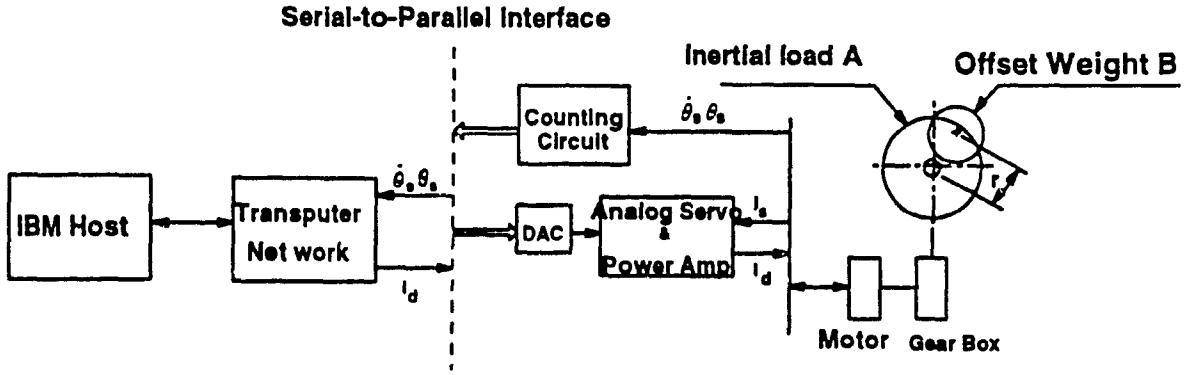


Fig. A.1 Experiment Set Up
for the model-based control of a simple mechanical load

The host computer is a IBM PC which functions as the operator's console, and the host of the transputer network. The transputer network is composed of three T800

transputers which integrates a 32-bit microprocessor, a 64-bit floating point unit, four standard transputer communication links, 4 Kbytes of on-chip RAM, a memory interface and peripheral interfacing on a single chip, using a 1.5 micron CMOS process [47]. The s/p interface [77] is designed to convert data streams between the transputer serial link and the devices like DAC, decoder to the position sensor etc.

With the system set up, sampling rate about 2500 Hz is achieved.

A.3 Computed Torque Control

The manipulator control problem involves the computation of the joint torques/forces that must be applied in order to make the manipulator track the desired trajectory. The dynamical model of the concerned motor_mechanical set up is derived by the Lagrangian formulation [11], and described by the following equation.

$$\tau = J\ddot{\theta} + G\cos\theta + \tau_f \qquad \textbf{(A.1)}$$

where

τ is the joint torque,

J is the inertia moment of the link,

G is the torque amplitude due to the gravity of the offset weight **B**,

θ is the displacement of the joint,

τ_f is the friction torque caused by the drive system. And it is composed of two pats: **Coulomb** and **viscous** friction, which is described by the following equation. where c_1, c_2, and c_3 are experimentally determined constants, and listed in Table A.1.

$$\tau_f = c_1\, sign\dot{\theta} + c_2\dot{\theta}$$
$$with\ the\ constraint \quad \max|c_2\dot{\theta}| = c_3$$

The control torque is calculated by the following equation.

$$\tau^* = \hat{J}[\ddot{\theta}_d + k_v(\dot{\theta}_d - \dot{\theta}) + k_p(\theta_d - \theta)] + \hat{G}\cos\theta + \hat{\tau}_f \qquad \textbf{(A.2)}$$

where

k_v and k_p are the derivative and the proportional gain respectively. And "^" indicates that the estimated system dynamics model is used in the computation.

Let ΔJ, ΔG and $\Delta\tau_f$ denote the errors in the estimates of the above dynamical parameters, i. e.

$$\begin{aligned} J &= \hat{J} + \Delta J \\ G &= \hat{G} + \Delta G \\ \tau_f &= \hat{\tau}_f + \Delta\tau_f \end{aligned} \qquad \textbf{(A.3)}$$

The control law of Equation (A.2) in conjunction with the dynamic model of Equation (A.1) yields the following error equation:

$$\ddot{e} + (1 - \Delta J J^{-1})(k_v\dot{e} + k_p e) = J^{-1}(\Delta\tau_f + \Delta J\ddot{\theta}_d) \qquad \textbf{(A.4)}$$

where $e = \theta_d - \theta$ is the position error of the joint.

And We could see clearly from the equation above that when the estimates of the dynamic model is exact, the error equation will reduce to a linear second order equation expressed by equation (A.5).

$$\ddot{e} + k_v\dot{e} + k_p e = 0 \qquad \textbf{(A.5)}$$

If there is no noise and initial errors, the manipulator will follow the desired

trajectory exactly. And if there is some noise or initial errors, the errors will be suppressed according to Equation (A.5).

The characteristic equation of the overall closed-loop system will be

$$s^2+k_v s+k_p=0 \tag{A.6}$$

The above equation could be expressed as

$$s^2+2\zeta\omega_n s+\omega_n^2=0 \tag{A.7}$$

Comparing Equation (A.6) and Equation (A.7), we get

$$\begin{aligned} k_p &= \omega_n^2 \\ k_v &= 2\zeta\omega_n \end{aligned} \tag{A.8}$$

where ζ and ω_n specify the damping ratio and natural frequency for the desired response. Selecting $\zeta = 1$ to get a critically damped system, then k_p and k_v should be determined by

$$\begin{aligned} k_p &= \omega_n^2 \\ k_v &= 2\omega_n \end{aligned} \tag{A.9}$$

where ω_n is the designed natural frequency of the system, which is determined experimentally as 105, Equation (A.9) gives $k_p = 11025$ and $k_v = 210$. The estimates of the parameters in the dynamic model of the robot are listed in table A.1.

Table A.1 Parameters in the Dynamic Model of a Simple Mechanical Load

J (kgm^2)	G (N-m)	Torque cons.(k) (N-m)/Amp	Gear ratio(N)	c_1 (N-m)	c_2 (N-m)	c_3 (N-m)
0.154	0.616	0.0485	20.4	1.0	0.6×10^{-3}	0.37

A.4 Experimental Results and Discussion

A.4.1 Trajectory Selection and Performance Evaluation Criteria

The desired trajectory, velocity, and acceleration is presented in Figure A.2, A.3, and A.4 respectively. It is composed of a fifth order polynomial and its reflection. The joint moves 8 complete turns in 50 seconds, and then comes back to its initial position in another 50 seconds.

For evaluating the performance of the robot control scheme, the peak trajectory tracking errors, the root mean square of the error (RMS), in terms of both position and velocity, are considered.

A.4.2 Tracking Performance

Presented in Figure A.5, A.6, and A.7 are the trajectory position, velocity errors, and the commanded torque respectively. The maximum position error is around 0.0098 radians. The RMS value of the position error is around 0.007 radians. The maximum velocity error is around 0.035 rad/sec. The RMS value of the velocity error is around 0.018 rad/sec. The corresponding sampling rate is around 2500 Hz. The error is mainly caused by the errors of the estimate of the parameters in the dynamic model. But the above experimental results are really a satisfactory as the system parameters are not measured systematically. Also an experiment was done using the same control law without consideration of the gravity force of the link, it is found that without dynamic compensation of the gravity force of the mechanical set up, the trajectory tracking performance of the mechanical set up is absolutely unacceptable.

A.4.3 Effect of the Sampling Rate on Tracking Performance

Sampling rate issue has been emphasized in [28, 40] and indicated in [3]. The sampling rate is paramount to the values of the controller gains. The higher the controller gains, the higher the stiffness of the system, which implies accurate

tracking performance. The experimental study of the effect of the sampling rate on the tracking performance will provide us a guidance for controller hardware design.

Shown in Figure A.8 is the maximum tracking position error, the root mean square of the position error, and the final position error versus the sampling rate. It could be seen that high tracking performance could be achieved by using a high sampling rate, but after about 300 Hz, the performance improvement is only marginally accomplished by the increase of the sampling rate. Further improvement of the tracking performance is only achievable by accuracy improvement to the dynamic model of the robot. The maximum tracking velocity error, the root mean square of the velocity error versus the sampling rate are depicted in Figure A.9.

A.4.4 Effect of the Speed of the Trajectory on Performance

By changing the displacement of the desired trajectory to different numbers and keeping the time 100 seconds as constant in the above described desired trajectory, different fastness of the trajectory is obtained. The fastness of the desired trajectory is indicated by the maximum speed along the desired trajectory. The effect of the fastness of the trajectory on the tracking performance is depicted in Figure A.10. The maximum speed along the trajectory is changed from 0.5 radians which is quite slow, to 5.65 radians which is very high from the practical operation of the real industrial robots. Compared to the rate of the change of the maximum speed with that of the performance degradation, little degradation of the performance is noticed, especially in terms of the RMS values both of the position errors and the velocity errors, which showed the effective dynamics compensation of the computed torque control.

A.5 Summary

To summarize this appendix, an experiment set up, experiment results of the computed torque control of a simple mechanical load is presented in a considerable detail. A dc motor coupled with a gear box is used to simulate a joint driving system

of a robot, while the mechanical load connected to the shaft of the gear box is used to simulate one link of a robot. The experiment verified the feasibility of the proposed controller hierarchy in chapter 3.

The study of the effect of the sampling rate of the computed torque control on the tracking performance showed that high tracking performance could be achieved by using a high sampling rate, but after the sampling rate reaches a certain value, the performance improvement of the tracking performance is only marginally accomplished by the increase of the sampling rate. Further improvement of the tracking performance is only achievable by accuracy improvement to the dynamic model of the controlled object.

The effect of the fastness of the trajectory on the tracking performance showed that compared to the rate of change of the maximum speed with that of the performance degradation, little degradation of the performance is noticed, especially in terms of the RMS values of the tracking errors, which showed the effective dynamic compensation of the computed torque control.

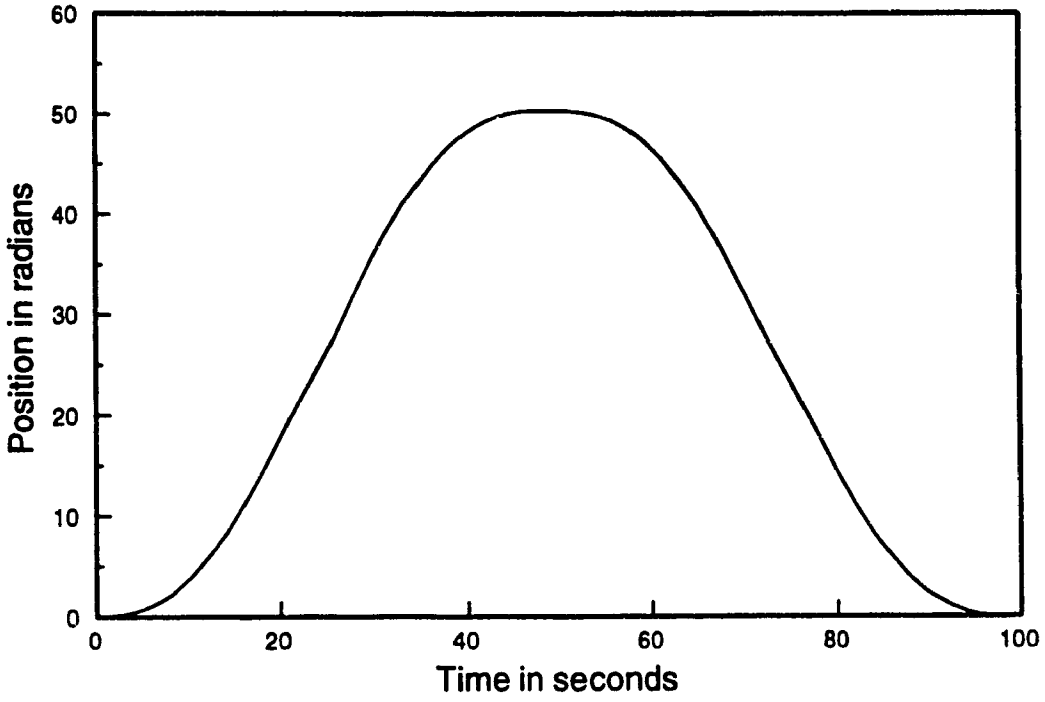


Fig. A.2 Desired Trajectory

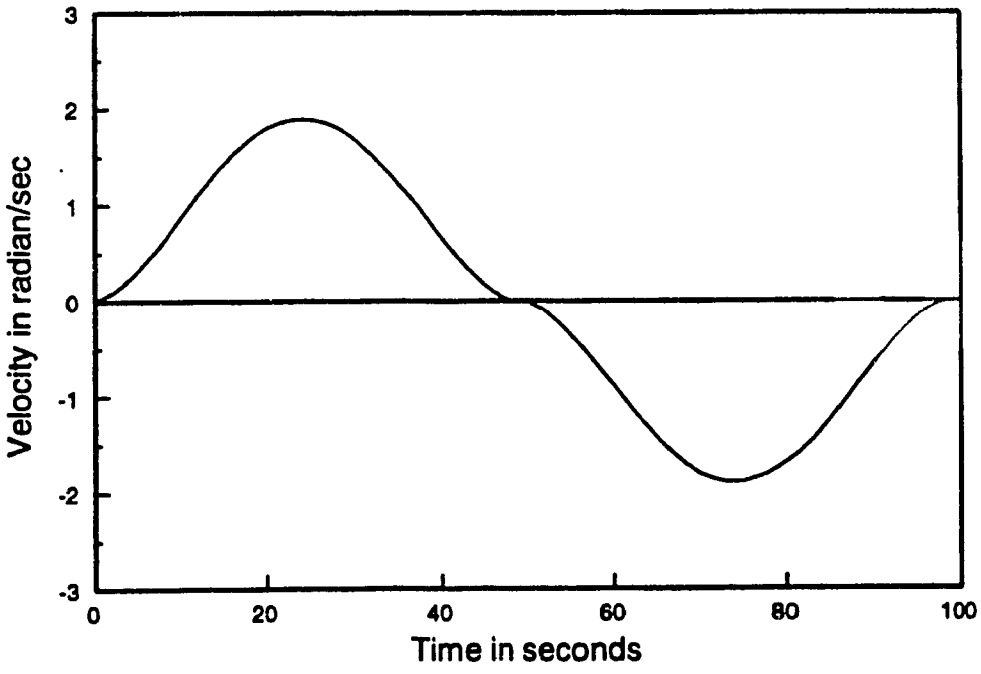


Fig. A.3 Desired Velocity

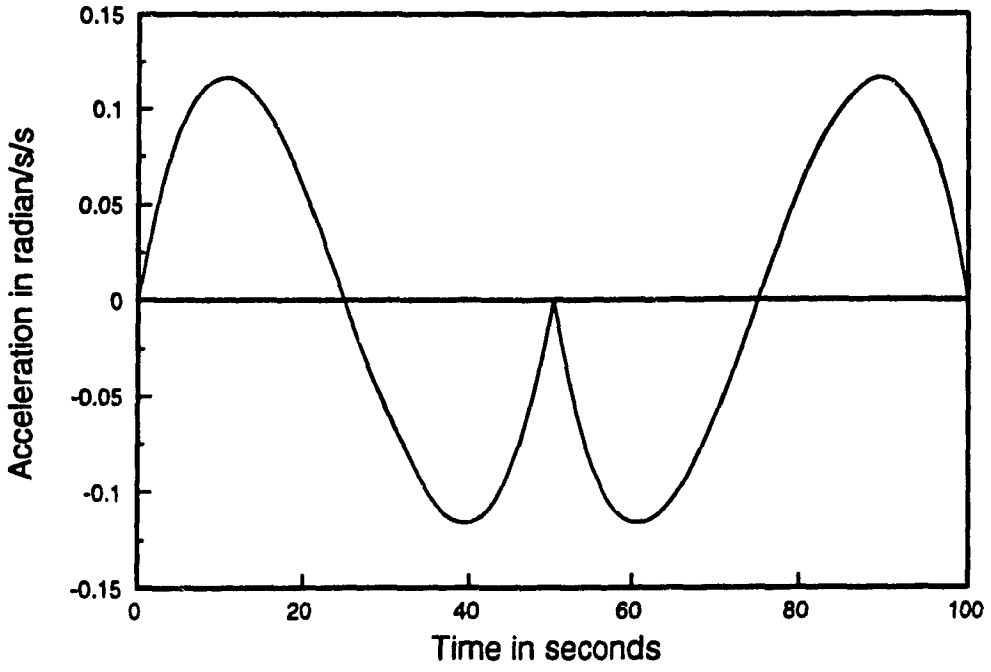


Fig. A.4 Desired Acceleration

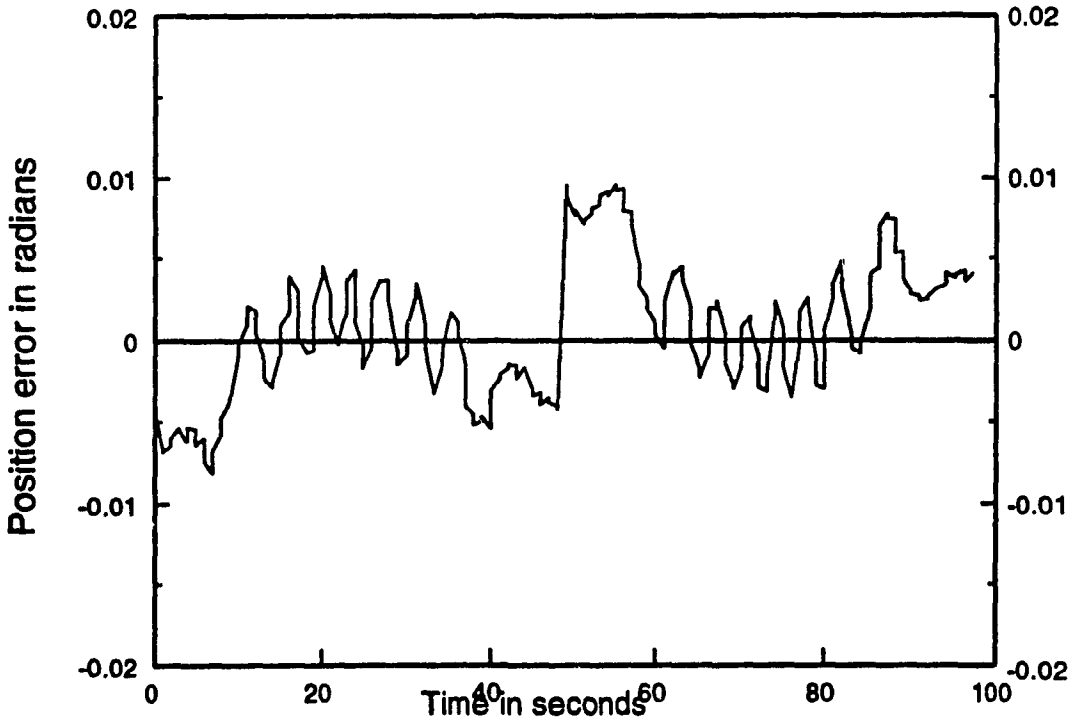


Fig. A.5 Position Tracking Error

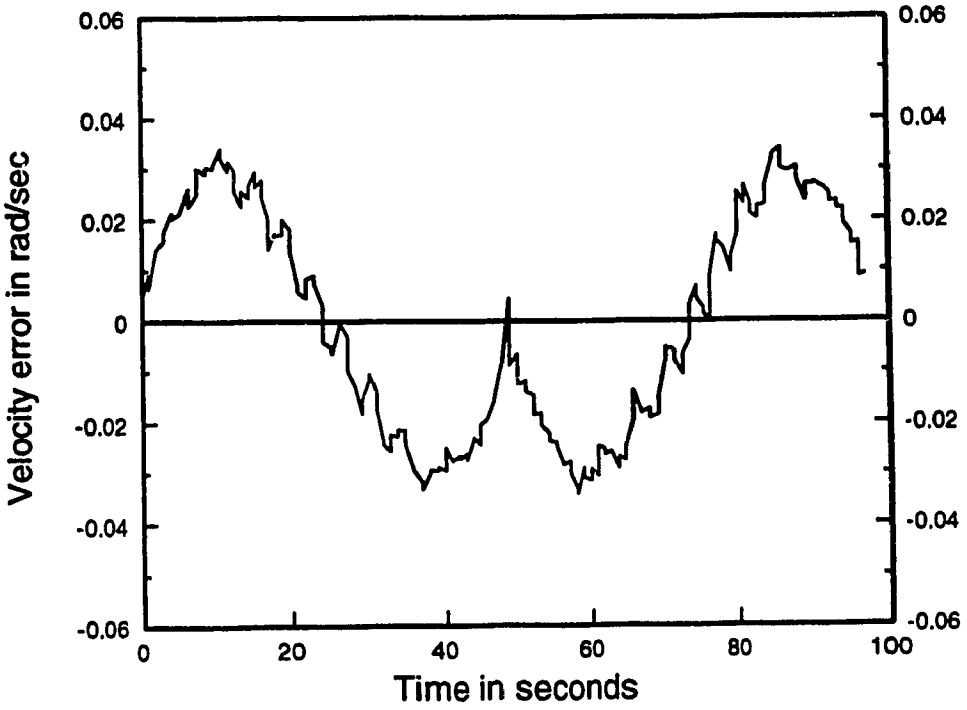


Fig. A.6 Velocity Tracking Error

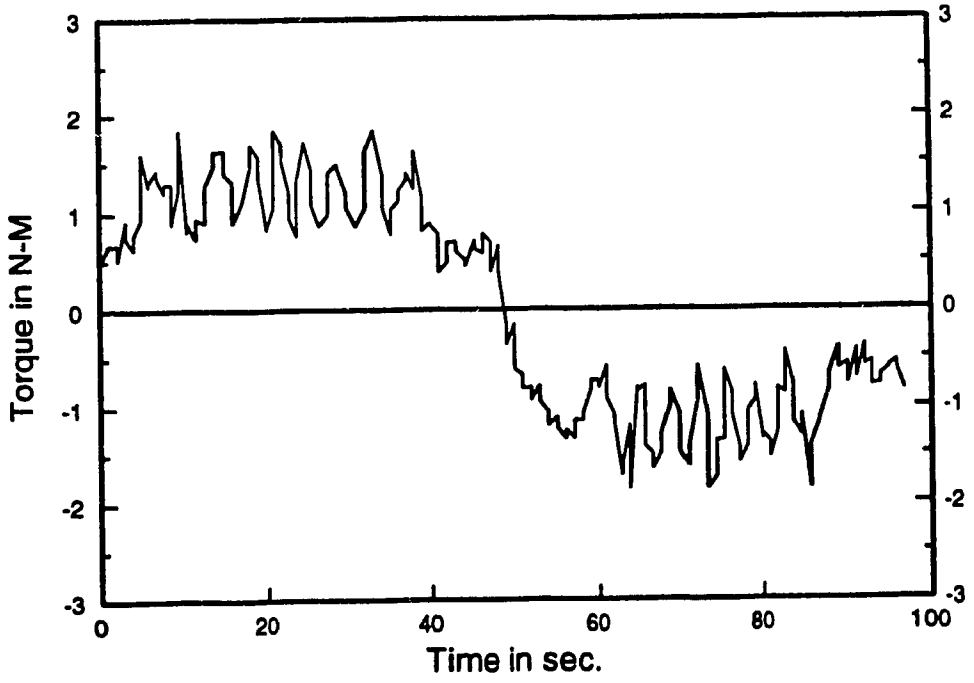


Fig. A.7 The Command Torque History

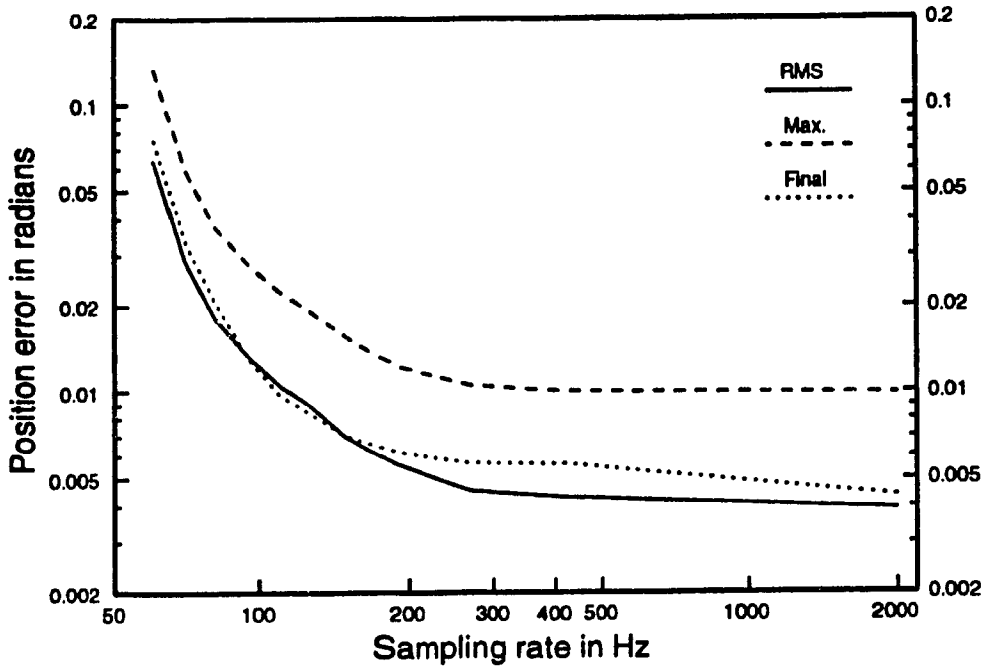


Fig. A.8 Effect of Sampling Rate on Performance

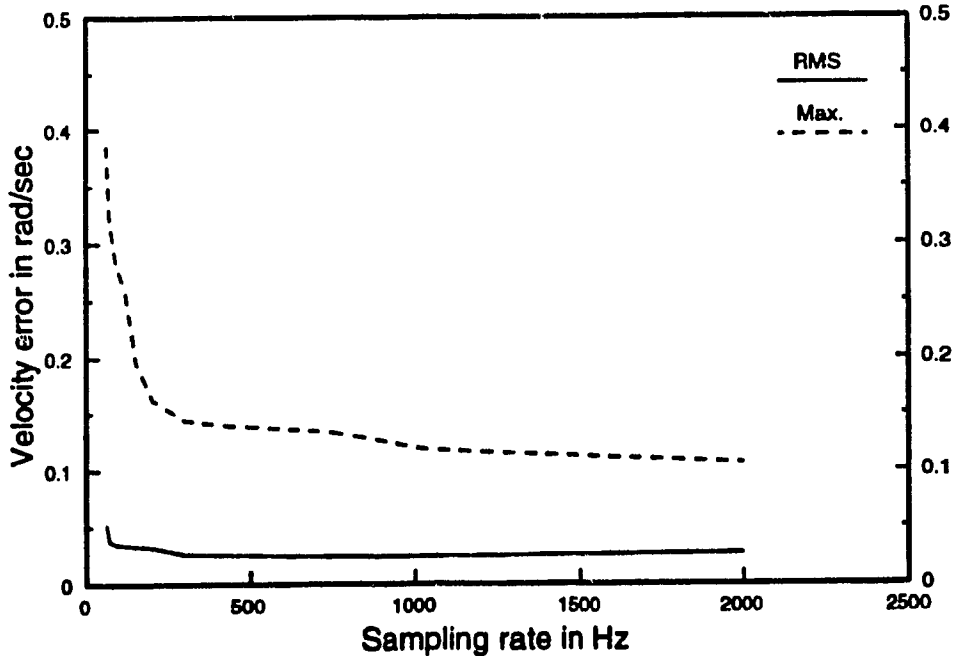


Fig. A.9 Effect of Sampling Rate on Velocity Tracking

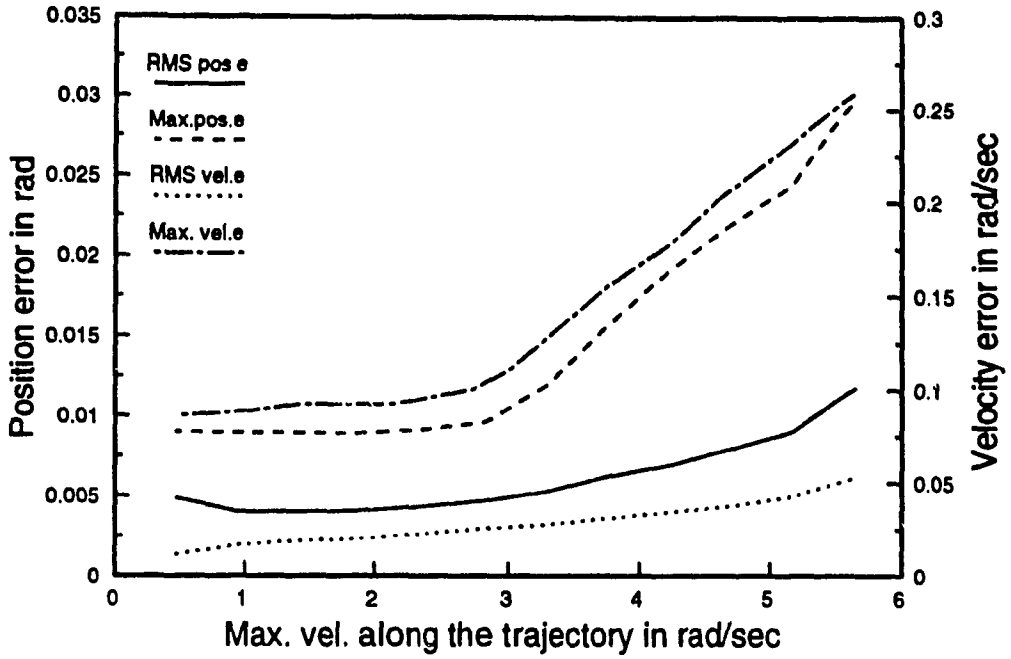


Fig. A.10 Effect of Speed of the Trajectory on the Tracking Performance

Appendix B

Developing Concurrent Applications Based on Transputers

B.1 Transputer : An Effective Element for Parallel Architectures

The **Transputer** [47] is a microcontroller with its own local memory and with links for connecting one transputer to another transputer. A transputer can be used in a single processor system or in a network to build high performance concurrent systems. In a network, the transputers communicate through the links resulting in a point to point communication.

The transputer architecture defines a family of programmable VLSI components. A typical member of the family is a single chip containing processor, memory and four communication links. In addition, each transputer product contains special circuitry and interfaces such as floating point unit (FPU), graphics processor etc., adapting it to a particular use. We discuss one of the device of this family T800 transputer in greater details.

The T800 transputer forms the general purpose node in the computed torque control computer architecture. Figure B.1 shows a block diagram of the T800. It has a 32 bit CPU with an 64 bit FPU on-chip. The FPU runs concurrently with the CPU, giving a sustained rate of 2.2 million floating point operations per second (MFlops), at

a clock speed of 20 MHz. There is 4Kbytes on-chip RAM for high speed processing. Four serial links are available for communication with other processors. At a link speed of 20Mbits/sec, a bidirectional data transfer rate of 2.4 Mbytes/sec per link is achieved. The CPU can address a linear address space of 4Gbytes.

The instruction set of the transputer is simple and efficient. About 70% of executed instructions are encoded in a single byte giving a peak rate of 20 million instructions per second (MIPS). The T800 has multi-tasking capabilities. The software model of the processor consists of several processes running in parallel. Typically, a process is a sequence of instructions which starts, performs a number of operations and is either suspended for reasons mentioned below or terminates to completion [78]. The T800 supports two levels of priority. Priority 1 (NON URGENT) processes are executed whenever there are no Priority 0 (URGENT) processes ready to run.

The processor has a microcoded scheduler which enables any number of processes to share the processor time. This removes the need for software kernel. At any time, a process may be :

ACTIVE - Being Executed

- On a waiting list to be executed.

INACTIVE - Waiting for an input.

- Waiting for an output.

- Waiting until a specified time.

The schedular operates in such a way that the **inactive process do not consume any processor time**. The URGENT process scheduling is non-preemptive i.e. the process

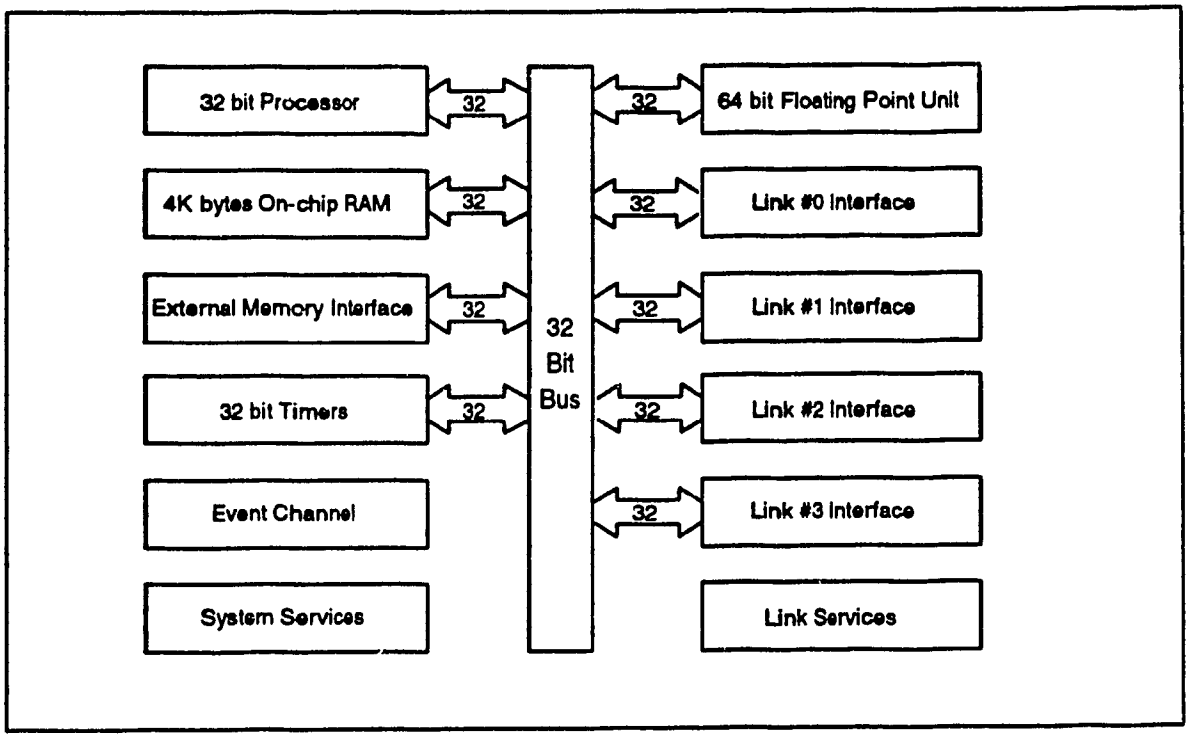


Fig. B.1 T800 Transputer Model

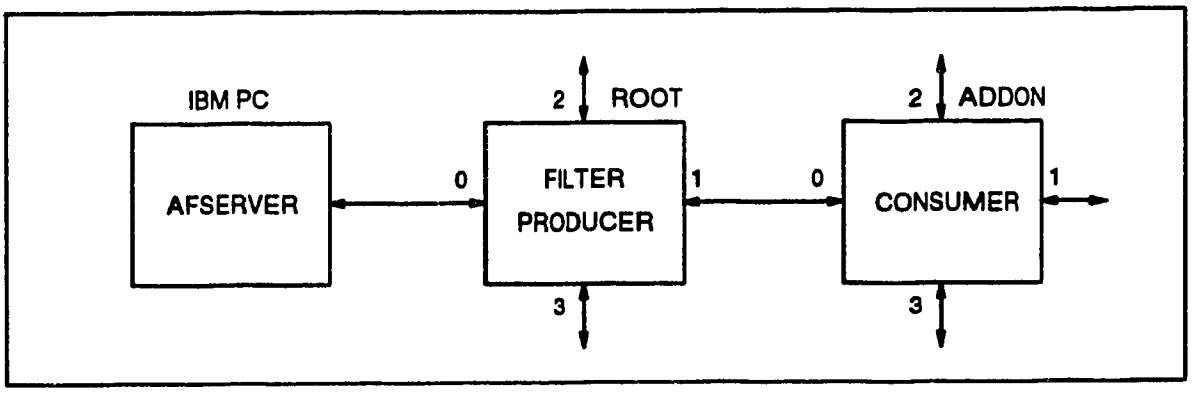


Fig. B.1 Two Transputer Implementation of Producer-Consumer Problem

will continue to execute till in the active state. In contrast, the NON-URGENT processes are periodically time sliced to provide an even distribution of processor time between them. Each time slice lasts for about 1 msec.

Communication between processes is achieved by means of channels. A channel between two processes on the same transputer is implemented by a single word in memory, a channel between processes executing on different transputers is implemented through the serial links discussed before. Process communication is point-to-point, synchronised and unbuffered. As a result, the channels need no process queue, no message queue and no message buffers. This leads to a communicating sequential process [79] software model for the transputer.

The T800 transputer also has two 32-bit timers, which can be used for a incorporating delays etc. One timer is accessible only to URGENT processes and is incremented every microsecond. The other is accessible only to NON URGENT processes and is incremented every 64 microseconds.

An Event channel is also provided for interrupting the transputer. If, one and only one, URGENT process is waiting for an input on this channel, then the interrupt latency (from when the channel becomes ready to the when the process starts executing) is typically less than 1 microsecond. The reader is referred to T800 data sheet [47] for further details on these features.

To summarise, one can say that the T800 transputer has all the features required for building a distributed memory multiprocessor system for the controller.

B.2 Development Environment

This section discusses other available transputer family of products, which are used for the development of the computed torque controller.

B.2.1 Hardware Support Tools

The following transputer hardware products are of interest in the computed torque controller implementation.

B.2.1.1 Transputer Module

The Transputer Modeles (TRAMs) are board level transputer products with simple and standardised interfaces [80]. Each TRAM has a processor interfaced to 1-8 Mbytes of RAM, four serial links for interprocessor communication and subsystem control circuitry. It is exceedingly compact and roughly the size of a credit card. A number of TRAMs can be interconnected to form multi transputer network.

A TRAM with T800 transputer interfaced to 1-2 Mbytes of memory suffices for a general purpose node of the network. Special nodes optimised for image processing, sensor interface etc. can be implemented using the following application specific TRAMs, available off the shelf.

(i) Framegrabber TRAM for real-time image capture and processing.

(ii) High Resolution Graphics TRAM, which supports upto 1024 x 1024 8 bit pixels, for enhanced graphics capabilities.

(iii) A to D Converter TRAM with 16 bit resolution and 200 KHz sampling rate for sensor interface.

Some limitations of the TRAM architecture are :

(i) The transputer data/address bus is not accessible for peripheral interface.

(ii) The Event channels is not accessible and can not be used to interrupt the processor.

B.2.1.2 Motherboard

A number of TRAMs are seated on a transputer motherboard to form a multi-transputer network. The motherboard is then plugged onto the host computer bus. Motherboards are available for IBM PC, PS/2, VME bus and SUN computer systems. One such motherboard, TM-B08 for IBM PC can accommodate 10 TRAMs, the transputer in slot 0, which is connected to the host is called the root transputer. Only the root transputer has the privilege to communicate with the host. Other transputer requests for host processor are routed through the root. Using DMA techniques, a data transfer rate of 200Kbytes to 300 Kbytes between the root transputer and the host can be established. Multiple motherboards can be interconnected in a straightforward manner to form large networks, as discussed in [80].

B.2.1.3 IMSC011

IMSC011 [78] link adaptor converts data streams between the serial transputer link and an eight bit parallel interface.

B.2.2 Software Development Tools

A number of software tools are available for developing applications on transputers. We start by considering the programming languages.

B.2.2.1 Programming Languages

To implement the parallelism possible in an algorithm, many concurrent languages have been developed over the last few years. Noticeably among them is the OCCAM programming language [81], which can be called as the assembly language for transputers. The OCCAM language bears a special relationship with the transputer architecture. Parallel computer systems can be developed in OCCAM and then implemented using transputers as "hardware OCCAM processes". However, OCCAM provides limited support for pointer operations, data structures and dynamic allocation. Also, OCCAM requires the use of a Transputer Development System for software development.

Other than OCCAM, parallel versions of many conventional languages have been developed such as Parallel C, Parallel Fortran and Parallel Pascal. The compilers for these are available from several vendors such as Logical Systems, Inmos and 3L etc. Taking into consideration the popularity of C language, its popularity and the available software support the Parallel C from 3L was adopted as the programming language for the software system development.

Parallel C has all the essential features of ANSI C along with the following

special function libraries to support concurrent programming :

(i) *Channels* <*channel.h*> : The basic communication primitives, *chan_in_message and chan_out_message*, for transferring a message across a channel are provided. The channel may be between two processes running on the same processor or between two tasks running on different transputers. Variations of basic primitive are provided for a byte transfer, word transfer or an arbitrary message length.

(ii) *Threads* <*thread.h*> : A set of functions such as *thread_create, thread_stop* etc. are provided to create new processes at run time. Threads, which are Parallel C equivalent of Unix process, are discussed in greater detail in section B.3.

(iii) *Semaphores* <*sema.h*> : A set of functions such as *sema_wait, sema_signal* etc., are provided to create, initialise and manipulate semaphores. Semaphores are used to synchronize the activity of several concurrently executing threads [78].

(iv) *Alt* <*alt.h*> : The *alt* (for alternative) functions allow a process to input from a group of channels, whichever becomes ready to communicate first. It provides a convenient means of handling external and interval events that must be handled by assembly level interrupt programming in conventional processors.

Similar functions for implementing *monitors, signals* etc., are available and the reader is referred to [51] for further details.

B.2.2.2 Debugging Tools

The availability of a powerful debugger can not be over stressed for projects requiring significant software development. TBUG [82] is a window based interactive

source level debugger available for Parallel C. Using TBUG one can interact with running transputer programs, which may contain several concurrent threads/tasks, by inserting break points, analyzing/modifying variable or memory etc. However, a faulty concurrent program with ill defined timing dependencies may produce unexpected results under TBUG, where such time relations get changed, making fault identification extremely difficult.

A last observation regarding the prices of the transputer products is in order. With increasing applications the prices of transputer products have been falling steadily over the last few years. As an example, a package consisting of a TMB08 motherboard for PC with five T800-20 (20 MHz clock) transputers, having upto six megabytes of memory, along with the Parallel C compiler for software development was available for as low as $4800, in the last quarter of 1990.

B.3 Developing Concurrent Applications on Transputers

This section briefly illustrates the techniques used in developing concurrent applications in Parallel C on a network of transputers. As an example, we consider the *producer-consumer* problem [78]. One program, called the producer, generates information which is to be used by the consumer program. With multi-tasking capabilities of the transputer, the producer-consumer problem can be implemented in two ways.

Multi-Thread Implementation : Parallelism on one processor

As mentioned before, Parallel C allows one to develop multi-thread applications. This means that a task (discussed below) can contain any number of concurrent

processes, running on the same processor, each of which is independently executing the code of the task. Although all the threads created by a task share their code and data memory, each can still operate independently because each thread has its own stack and instruction pointer.

The box on the next page shows an implementation of producer-consumer problem in Parallel C using channels for synchronisation. Recall that a channel in this case is just a shared memory location where the producer leaves its result and the consumer retrieves it in a synchronised fashion. The same could have been implemented by using a shared data buffer between the two threads, access to which is interlocked using semaphores.

Multi-Task Implementation : Parallelism on Transputer Networks

If the producer and consumer programs are computationally intensive, processing time can be reduced substantially by mapping them on different transputers as independent tasks. A task in Parallel C has its own code and data areas separate from all other tasks, even when several of them are placed on the same processor. As there is no shared memory between tasks, they only communicate via channels.

For implementation as a multi-tasking application, a configuration file to describe hardware and software configuration needs to be defined. A configuration language and a software utility called configurer are provided in Parallel C towards this purpose. For a two transputer implementation of producer-consumer, as shown in Figure B.2, the required configuration file is as shown on the next page.

The afserver task is vendor supplied server which runs on the host and caters to

```
#include <thread.h>
#include <chan.h>

#define        STACKSIZE   1024

CHAN chan,consumer_finished;

void   producer()     /* To generate 10 values */
{
       int i;
       for (i=0;i<10;i++)
              chan_out_word(i,&chan);
}

void   consumer()     /* To consume 10 values */
{
       int i,val;
       for (i=0;i<10;i++)
              chan_in_word(&val,&chan);

       chan_out_word(1, &consumer_finished);
}

main()
{
       int dummy;

       chan_init(&chan);              /* Initialise channels */
       chan_init(&consumer_finished);

       thread_create(producer, STACKSIZE, 0);
       thread_create(consumer, STACKSIZE, 0);

       /* Wait for all threads to terminate  before exiting */
       chan_in_word(&dummy, &consumer_finished);
}
```

Multi-Thread Implementation of Producer-Consumer problem

the transputer request for host services such as file I/O, peripheral access etc. The *filter* task is also vendor supplied and is required because of some communication problems associated with T414A transputer in particular.

```
!       Producerconsumer.cfg

processor     host          !The PC
processor     root
processor     addon

wire    root[0]-host[0]     !Link connection between root Link 0 and host
wire    root[1]-addon[0]    !Link connection between Producer and Consumer

task    afserver ins=1 outs=1
task    filter     ins=2 outs=2 data=10K
task    producer ins=2 outs=2
task    consumer ins=1 outs=1

place   afserver host       !afserver runs on the PC
place   filter     root     !filter runs on the root
place   producer root               !producer also runs on the root
place   consumer addon      !consumer runs on addon

                            !Establish two-way connections between tasks
connect ? afserver[0] filter[0]
connect ? filter[0] afserver[0]
connect ? filter[1] producer[0]
connect ? producer[0] filter[1]
connect ? producer[1] consumer[0]
connect ? consumer[0] producer[1]
```

Configuration File for Two-Transputer Implementation

The modified code for the two programs for multi-task application is shown on the above box. The channel used in this case is mapped over the physical link connecting the two processors.

B.4 Performance of Transputers in Real Time Control

Apart from robotics, the use of transputers in other real-time applications is becoming increasingly popular. Asher et.al. [83] have used transputers for high

```
        /* Producer */
#include <chan.h>

main(int argc, char *argv[], char
*envp[],      CHAN *in_port[], int ins,
CHAN          *out_ports[], int outs)

{
int    i;

for (i=0;i<10;i++)
       chan_out_word(i, out_ports[1]);

}
```

```
        /* Consumer */
#include <chan.h>

main(int argc, char &argv[], char
*envp[], CHAN *in_ports[], int    ins,
CHAN *out_ports[], int outs)

{
int i,val;

for(i=0;i<10;i++)
       chan_in_word(&val, in_ports[0]);

}
```

Multi-Task Implementation of Producer-Consumer problem

performance vector control of AC induction motors. Thompsom et. al. [84] have built a controller for gas-turbine engines using transputers and incorporating hardware/software redundancy for fault tolerance.

In the field of robotics, Whitcomb et. al. [85] have built a transputer based test bed for manipulator control algorithms. However, the set up is not cost effective as a dedicated transputer is used for each joint. Sharkey et.al. [40] have used a pipeline of six T800 for variable structure control and inverse dynamics of Puma 560. Hashimoto et.al. [35] have used parallelism in Newton-Euler algorithm for inverse dynamics to implemented the computed torque control for the first three links of Puma 560. Using a T800 processor for each joint, they achieve a sampling interval of 0.66 msec.

S. Kabra [6], has reported that it takes **5.824 milleseconds** and **1.344 milleseconds** for the purely sequential implementation of forward and inverse kinematics on one T800 transputer respectively. Xiao et. al. [9] have achieved a time of **0.8** msec

for inverse dynamics algorithm of Puma 560 on three transputers.

B.5 Summary

The description and discussion of transputers has shown that, Transputer, with its unique architecture of local memory and four serial links, is found to be an appropriate device for the construction of the computer structure of the model-based controller. The development tools provided both in the software and hardware aspect, make the real time development easier. The development of concurrent applications based on transputers is discussed.

Appendix C

Device Addresses

Symbol	Address	Device
COUNTER1	0x01	Counter for Joint #1 (Base).
COUNTER2	0x02	Counter for Joint #2.
COUNTER3	0x03	Counter for Joint #3.
COUNTER4	0x04	Counter for Joint #4.
COUNTER5	0x05	Counter for Joint #5.
COUNTER6	0x06	Counter for Joint #6.
ALL_COUNTER	0x0f	Counter for Joint #1 to #6.
AD1_ADD	0x0c	A/D Converter #1 (Eight Channels).
AD2_ADD	0x0d	A/D Converter #2 (Eight Channels).
HND_PORT	0x07	Gripper Controller.
PORT_40V	0x0a	40 Volt status port.
DA1_ADD	0x11	D/A Converter #1 (Eight Channels).
DA2_ADD	0x12	D/A Converter #2 (Eight Channels).
DA3_ADD	0x13	D/A Converter #3 (Eight Channels).
DA4_ADD	0x14	D/A Converter #4 (Eight Channels).
DA5_ADD	0x15	D/A Converter #5 (Eight Channels).
DA6_ADD	0x16	D/A Converter #6 (Eight Channels).

Appendix D

D/A Converter Calibration

The relationship of the command current (I_c) and the digital value (the corresponding decimal format) sent to the DAC (DAC Input Value) for each joint of the robot is calibrated by sending a sequential of binary data from 00H to FFH to the DAC, measuring the real current through the armature circuitry of the d.c. motor. A linear relationship is found and listed bellow

Joint 1: $DAC\ Input\ Value = 21.1292 \times I_c + 129.0788$

Joint 2: $DAC\ Input\ Value = 22.1649 \times I_c + 129.8544$

Joint 3: $DAC\ Input\ Value = 21.4167 \times I_c + 129.2429$

Joint 4: $DAC\ Input\ Value = 41.3623 \times I_c + 130.0887$

Joint 5: $DAC\ Input\ Value = 41.6969 \times I_c + 127.6493$

Joint 6: $DAC\ Input\ Value = 43.2848 \times I_c + 128.4405$

Appendix E

PUMA 560 Robot Arm Related Data

Table D.1 Puma 560 link paramemters.

Joint	α_i	a_i	d_i	Joint Range	Angle at READY
1	-90	0	0	$-160 \leq \theta_1 \leq 160$	0
2	0	431.80	149.09	$-180 \leq \theta_2 \leq 43$ $137 \leq \theta_2 \leq 180$	-90
3	90	-20.32	0	$-180 \leq \theta_3 \leq -128$ $-52 \leq \theta_3 \leq 180$	90
4	-90	0	433.07	$-110 \leq \theta_4 \leq 170$	0
5	90	0	0	$-100 \leq \theta_5 \leq 100$	0
6	0	0	56.25	$-266 \leq \theta_5 \leq 266$	0

Note : All dimensions in millimeter and all angles in degrees.

Note : Minimum average velocity is taken as 1count/65536T_s.

The joint angle to encoder count conversion is given by

$$\mathbf{e} = \mathbf{R}\theta$$

where θ is the joint angle vector in radians. **R** for Puma 560 is given by

$$R = \begin{bmatrix} 9964.87 & 0 & 0 & 0 & 0 & 0 \\ 0 & 13727.43 & 0 & 0 & 0 & 0 \\ 0 & 0 & 8547.62 & 0 & 0 & 0 \\ 0 & 0 & 0 & 12101.57 & 0 & 0 \\ 0 & 0 & 0 & 159.15 & 11446.92 & 0 \\ 0 & 0 & 0 & 159.26 & 2203.69 & 12205.02 \end{bmatrix} \quad \text{(D.1)}$$

The encoder count to joint angle conversion is given by

$$\theta = \mathbf{R}^{-1}\mathbf{e}$$

where $\mathbf{R}^{-1}$ is as follows

$$R^{-1} = \begin{bmatrix} 0.1004 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0.0728 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0.1170 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0.0826 & 0 & 0 \\ 0 & 0 & 0 & -0.0011 & 0.0874 & 0 \\ 0 & 0 & 0 & -0.0009 & -0.0158 & 0.0819 \end{bmatrix} x10^{-3} \qquad (D.2)$$

Table D.2 Puma 560 Servo Resolution

Joint	Encoder Lines	Gear Ratio	Angle Resolution (degree)	Instantaneous Velocity Resolution (degree/sec)	Minimum Average Velocity (degree/sec)
1	250	62.6111	5.752e-3	22.47	3.428e-4
2	200	107.815	4.171e-3	16.29	2.486e-4
3	250	53.7063	6.703e-3	26.18	3.995e-4
4	250	76.0364	4.732e-3	18.48	2.820e-4
5	250	71.9231	5.007e-3	19.56	2.984e-4
6	250	76.6864	4.692e-3	18.33	2.796e-4

Appendix F

Parameters in the Dynamical Model of Puma 560 Robot

Table F.1 Link Mass and Centres of Gravity

Link No.	Mass(kg)	r_x(m)	r_y(m)	r_z(m)
Link 2	17.40	0.068	0.006	-0.016
Link 3	4.80	0	-0.070	0.014
Link 3 with wrist	6.02	0	-0.143	0.014
Link 4	0.82	0	0	-0.019
Link 5	0.34	0	0	0
Link 6	0.09	0	0	0.032
Wrist	1.24	0	0	-0.064

Table F.2 Inertia Dyadics and Effective Motor Inertia

Link No.	I_{xx}	I_{yy}	I_{zz}	I_{motor}
Link 1	-	-	0.35	1.14($\pm$0.27)
Link 2	0.13($\pm$0.01)	0.524($\pm$0.05)	0.539($\pm$0.03)	4.71($\pm$0.54)
Link 3	0.066	0.0125	0.086	0.83($\pm$0.09)
Link 3 with Wrist	0.19($\pm$0.04)	0.015($\pm$0.05)	0.212($\pm$0.04)	-
Link 4	1.80×10^{-3}	1.80×10^{-3}	1.30×10^{-3}	0.200($\pm$0.016)
Link 5	$0.30x^{-3}$	0.30×10^{-3}	0.40×10^{-3}	0.179($\pm$0.014)
Link 6	0.15×10^{-3}	0.15×10^{-3}	0.04×10^{-3}	0.193($\pm$0.015)

Table F.3 Motor and Drive Parameters

Joint No.	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
Gear Ratio	62.61	107.36	53.69	76.01	71.91	76.73
K^{τ}(NM/A)	0.2611	0.2611	0.2611	0.0988	0.0988	0.0988
τ_c (NM)	5.95	6.82	3.91	1.07	0.89	0.94

Appendix G

Configuration File

(Refer to page 55)

```
!
! PUMA.CFG              Configuration file for model-based robot control
!
! Hardware
!
processor host
processor root          !Root transputer running FILTER
processor ioserv        !Additional transputer for MUX
processor master        !MASTER transputer running TASK CPP
processor tc560         !k for Puma560 runnung TASK kin560
processor ct560         !P for Puma 560 running TASK CTA560
processor cubspl        !Transputer running INTPOL

wire ? host[0] root[0]  !anonymous wire connecting PC to transputer

!Connections of MUX with other tasks for debugging and I/O
wire ? root[2] ioserv[1]    !Link between FILTER and MUX
wire ? ioserv[2] master[1]  !Link between MUX and CPP
wire ? ioserv[3] ct560[3]   !Link between MUX and CTA
wire ? ioserv[0] tc560[0]

!Connection between CPP and other tasks for control
wire ? master[0] cubspl[0]  !Link between CPP and INTPOL
wire ? master[2] ct560[1]   !Link between CPP and CTA

!Connection between TC and other tasks for control
wire ? ct560[2] tc560[1]    !Link between CTA and TC

!Connection between KIN560 and ITPOL
wire ? tc560[2] cubspl[1]   !Link between TC and INTPOL

!Links used for communicating with S/P interfaces
!ct560[0] - S/P Puma 560    !Link between CTA and S/P interface

!
! Task declarations indicating channel I/O ports and memory requirements
```

```
!
task afserver ins=1 outs=1
task filter ins=2 outs=2
task multiplexer file=filemux ins=4 outs=4
task cpp ins=4 outs=4
task kin560  ins=3 outs=3
task dyn560 ins=4 outs=4
task intpol ins=1 outs=1
!
! Assign software tasks to physical processors
!
place afserver host
place filter root
place multiplexer ioserv
place cpp       master
place kin560  tc560
place dyn560  ct560
place intpol   cubspl
!
! Set up the connections between the tasks.
!
connect ? filter[0] afserver[0]
connect ? afserver[0] filter[0]

connect ? filter[1] multiplexer[0]
connect ? multiplexer[0] filter[1]

connect ? multiplexer[1] cpp[1]
connect ? cpp[1] multiplexer[1]

connect ? multiplexer[2] dyn560[1]
connect ? dyn560[1] multiplexer[2]

connect ? multiplexer[3] kin560[1]
connect ? kin560[1] multiplexer[3]

connect ? cpp[2] intpol[0]
connect ? intpol[0] cpp[2]

connect ? cpp[3] dyn560[2]
connect ? dyn560[2] cpp[3]

connect ? kin560[2] dyn560[3]
connect ? dyn560[3] kin560[2]
```