AN APPROACH TO MOTION PLANNING AND MOTION CONTROL OF TWO ROBOTS IN A COMMON WORKSPACE

by

Bum Hee Lee

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy
(Computer, Information, and Control Engineering)
in The University of Michigan
1985

Doctoral Committee:

Assistant Professor C. S. George Lee, Chairman

Professor Elmer G. Gilbert

Professor Richard A. Volz

Professor Kensall D. Wise

Associate Professor Anthony C. Woo

ABSTRACT

AN APPROACH TO MOTION PLANNING AND MOTION CONTROL OF TWO ROBOTS IN A COMMON WORKSPACE

by

Bum Hee Lee

Chairman: C. S. George Lee

This thesis focuses on motion planning and motion control for two robots in a common workspace, which involves the investigation and development of an off-line discrete-time trajectory planning scheme of straight line paths in Cartesian space; off-line collision-free path planning algorithms for two robots; and an on-line resolved motion adaptive perturbation control method.

A discrete-time trajectory planning scheme for a straight line path has been developed to determine the trajectory set points in Cartesian coordinates which satisfy both smoothness and torque constraints. An iterative forward and backward search algorithm coupled with a modified forward search algorithm is developed to determine the joint values at each servo interval on the given straight line path. Relaxation points are needed in planning the trajectory to meet the boundary conditions at the final location of the straight line path. The resultant straight line trajectory is composed of three segments: an acceleration, a deceleration, and a relaxation portion.

Using a sphere model for the wrist of a manipulator, planning collision-free paths of two time-scheduled robots is considered. For planning a collision-free path

of a moving robot in the presence of a stationary robot, several performance measures are considered, from which two performance measures are selected and used for planning a collision-free path. For planning collision-free paths of two moving robots, various collision situations are identified. Notions of a collision map and time scheduling are introduced, which are used to develop two procedures for planning collision-free paths of the two moving robots.

An adaptive control in Cartesian space, which adopts the idea of resolved motion rate and acceleration control, has been developed to track the collision-free trajectory faithfully over a wide range of manipulator motion and payloads. The control scheme is based on the linearized perturbation equations along a desired hand trajectory. The control system is characterized by feedforward and feedback components which can be computed separately and simultaneously. A feasibility study of implementing the adaptive control for a six-joint PUMA manipulator is explored using present-day low-cost microprocessors.

Finally, the proposed straight line trajectory planning scheme is simulated on a VAX-11/780 computer to evaluate its performance.

To My Parents, Wife, and Children:

Yong Jei and Yong Sun

ACKNOWLEDGMENTS

It is a pleasure to express my appreciation to a number of people for their help during my graduate study.

I am most grateful to my advisor, Professor C. S. George Lee, for his constant guidance and encouragement without which this dissertation would not have been completed. His kindness, patience and invaluable ideas are so helpful that it has been a delight to work with him throughout my entire graduate study.

My appreciation goes to the members of my Dissertation Committee, Professor Gilbert, Professor Volz, Professor Wise, and Professor Woo for the time and efforts they have spent in reading, commenting on, and enhancing the quality of this dissertation. Especially, Professor Gilbert and Professor Volz have patiently reviewed and improved many revisions of this work. Finally, to my parents and wife, I wish to express my thanks for their endless love and encouragement during my graduate study.

During the last two years of this work, it was partially supported by the National Science Foundation Grant ECS-8106954 and the Robot Systems Division in the Center for Research in Integrated Manufacturing (CRIM) under the AFOSR Grant F49620-82-C-0089.

TABLE OF CONTENTS

LIST OF FIGURES

Figure

LIST OF TABLES

Table

LIST OF APPENDICES

CHAPTER 1

INTRODUCTION

1.1. Introduction

An industrial robot is a general purpose manipulator which consists of several rigid bodies (or links) connected in series by revolute or prismatic joints (see Figure 1.1 [24]). One end of the chain is attached to a supporting base while the other end is free and attached with a tool to manipulate objects or perform assembly tasks.

Mechanically a robot is composed of an arm (or main frame) and a wrist subassembly plus a tool. It is designed to reach a workpiece located within its work volume. The work volume is the sphere of influence such that the arm can deliver the wrist subassembly unit to any point within the sphere. The arm subassembly typically consists of three degrees of freedom movement. The combination of the movements of the arm subassembly will place or position the wrist unit at the workpiece. The wrist subassembly unit usually consists of three rotary motions. The combination of these motions will orient the tool according to the configuration of the object to ease pickup. Hence for a six-joint robot, the arm subassembly is the positioning mechanism, while the wrist subassembly is the orientation mechanism.

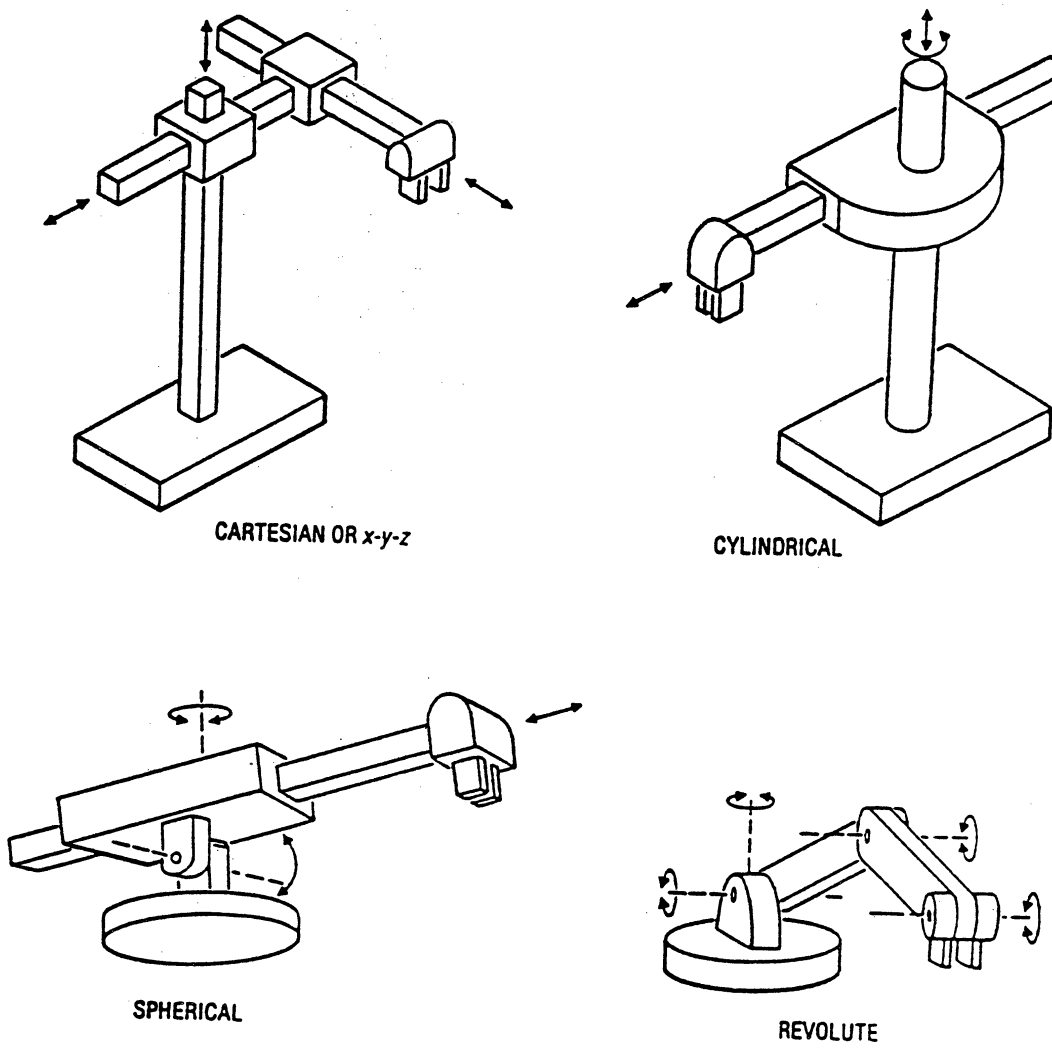


Figure 1.1 Configurations of Robots

Presently there are many commercially available industrial robots which are widely used in simple material-handling, spot/arc welding, and loading and unloading numerically-controlled machines. These robots exhibit their characteristics in motion and geometry. Mechanically they fall into one of the four basic motion-defining categories according to the motion of the arm subassembly (see Figure 1.1):

- Cartesian robot (three linear axes)
- Cylindrical robot (two linear and one rotary axes)
- Spherical robot (one linear and two rotary axes)
- Revolute or Articulated robot (three rotary axes)

Most automated manufacturing tasks are done by special purpose machines which are designed to perform their prespecified functions in a manufacturing process. The inflexibility of these machines makes the computer-controlled manipulators more attractive and cost effective in various manufacturing and assembly tasks. These include cost savings, reliability, tolerance of working environments unacceptable to humans, and an adaptability to both structured and unstructured environments through simple reprogramming. The end results are improved productivity, efficiency, and flexibility in industrial manufacturing and automation.

Due to the recent development of sophisticated sensors, advanced programming languages, and efficient visual recognition methods, the application domain of robots is being expanded to more sophisticated tasks and in particular assembly tasks [4,55,60]. Undoubtably, more robots will be put to use in assembly tasks in manufacturing cells which, as described by Whitney [61], are small size manufacturing systems in which there are sequences of coordinated operations among various manufacturing devices. These operations require the robot arms to move their end-

effectors from one location in the Cartesian space to another in coordinated motion without any collisions with other objects.

Current industrial practice employs simple time-space coordination which does not allow more than one robot working in a common workspace. Such coordination and control result in under-utilization of robot arm capabilities. Any improvements in the performance of controlling two robot arms in a common workspace will require further investigation of sophisticated straight line trajectory planning, collision avoidance strategies, and Cartesian space control strategies. This initiates and motivates the research efforts that constitute this thesis.

1.2. The Problem and The Approach

Undoubtably, the use of two robots in a common workspace will increase productivity. In this system configuration, given the locations of workpieces, the robot arms have to sense the workpieces, manipulate them and transfer the assembled workpieces to designated locations. However, with two robots working in a common workspace, problems are created that were not anticipated in a single stand-alone robot system.

While one robot tries to assist the other robot, it may become an "obstacle" to the other robot. Thus, before each robot executes its motion command, it is important for the motion planning system to detect the possibility of potential collisions between the wrists of the robots.[1] If the wrist of a robot is found to be on the path of the other robot, then paths must be replanned or the traveling time on the paths must be modified to avoid the potential collisions. Thus, the main task of this

[1]We assume in this system configuration that there are no collisions between the first three links of the robots.

thesis is to develop motion planning and motion control algorithms and methodologies so that two robots can work safely in a common workspace. This objective leads naturally to the investigation of the following three separate but coherent areas:

(1) Off-line planning of straight line manipulator trajectory

(2) Off-line planning of wrist collision-free path, and

(3) On-line Cartesian space control for tracking the wrist collision-free straight line trajectory

That is, we are concerned with the formalism of describing the desired manipulator motion as a sequence of set points in the Cartesian space (position and orientation of the manipulator hand) through which the manipulator hand must pass in straight line motion, must avoid any possible wrist collisions, and must be controlled with high accuracy.

It has been well recognized that optimum control of a manipulator is very difficult due to its inherent nonlinearity and complexity in the equations of motion, in addition to the nonlinear trigonometric function relationship between the joint coordinates and the Cartesian coordinates [24,30,38,46]. Thus, optimum control of a manipulator can be conveniently decomposed into an off-line trajectory planning (motion planning) followed by an on-line path tracking control scheme (motion control) [22]. This decomposition reduces the real time computational burden of the optimal controller and transfers the computational burden to the off-line trajectory planning phase. This thesis rests on a similar decomposition concept of manipulator control.

The motion planning phase focuses on the investigation of a straight line trajectory planning scheme in Cartesian space followed by wrist collision detection along the trajectory, and replanning the path or time-scheduling of the original trajectory if necessary. Several advantages exist for designing a straight line trajectory for controlling the manipulator:

(1) Visibility concept: If one of the manipulators cannot "see" its destination position, then obstacle(s) exists on the path.[2]

(2) Motion correctness: It is easier for the user to specify straight line Cartesian motion and the correctness of motion can be verified from the motion of the manipulator hand.

(3) Shortest path: The straight line path is the shortest path between two assembly locations even if the required traveling time may not be minimum.

(4) Time-varying obstacles: If two robots are not moving in straight line motion, it may be extremely difficult to handle the time-varying environments for mutual wrist collision avoidance.

(5) Path Information: After detecting that potential collisions exist on the path, we can utilize the path information of one robot to decide the path or trajectory of the other robot for collision-free motion.

Since the location of each robot on the straight line path must be available at all servo instant to detect the existence of potential collisions and to obtain a collision-free path in the two moving robot environment, a servo time based trajectory planner and a collision-free path generator will be the immediate problems to

[2]The reverse is not true. That is, if the manipulator can see the destination position, it still does not *guarantee the path to be collision free.*

be solved. An accurate Cartesian space control scheme must be used to follow the collision-free trajectories faithfully. Each of these areas will be covered in detail in Chapters 3, 4, and 5, respectively.

1.3. Literature Survey

In this section, a brief review of previous work on multirobot control systems, trajectory planning, collision avoidance, and adaptive control strategies is presented.

There are only a few research institutions working on multirobot control problems. Conventional techniques involve careful off-line trajectory planning such that the robot arms do not collide with known obstacles in fixed locations. This technique involves "interpolation" of intermediate Cartesian knot points that the manipulator hand must travel by a class of polynominal functions. Others are involved in the development of efficient sensory interactive hierarchical control system for the "factory of the future" [2,4,45,54,56,61].

At the National Bureau of Standards, a hierarchical control system with sensory feedback at various levels/stages coupled with the CMAC (Cerebellar Model Articulation Control) concept [2] has been proposed for manufacturing systems. The focus is mainly on the global control of "loosely-coupled" robots in a manufacturing environment. The two robot system in a common workspace is a "tightly-coupled" system which requires much more intensive computations and coordinations. Although, significant theoretical results about the multiple robot control problems were not obtained, it was found that, for controlling multiple robots, an exchange of information among various robots in real time is necessary to achieve collision avoidance.

Saridis [51] proposed an intelligent control concept for controlling multiple robots in an "optimized" manner using both artificial intelligence and control discipline ideas. This presents a desirable research direction for multiple robot systems.

An interesting modular system approach [54] with computer communication network between a supervisory computer and functional module computers is being investigated by researchers at SRI. Their approach is based on sensory feedback signals to actuate control strategies. They employed very simple algorithms to calculate hand positions to avoid potential collisions and emphasized vision feedback signal processing in global coordinate system for two robot arms.

Recently, Freund [16,17] exploited table look-up and hierarchical decisions to manage collision avoidance for two robot arms in a common workspace. His approach is based on the hierarchical kinematic decisions for positioning the manipulator links to avoid potential collisions. This technique focuses more on gross motion control of two robot arms than the fine motion control of two robot arms.

Although, the above approaches solved the problems in the multiple robot system with various degrees of success, they do not address the problems of the two robot system adequately for the following reasons: (1) most of the approaches are based on kinematic information, (2) most of the obstacles are fixed in location, (3) no collision avoidance is considered for the wrists (last three links of the robot) of the two robots. In view of this, the approach employed in this thesis is based on the off-line planning of a straight line path and trajectory with an on-line path tracking control strategy. This involves planning of the straight line trajectory in Cartesian space, appropriate geometric modeling of the wrists which will be used in detecting the potential wrist collisions along the path, replanning the path or time-

scheduling of the original trajectories if collisions exist, and finally controlling the robots to track the desired collision-free trajectories faithfully. We shall now review the previous work in the above areas, namely, trajectory planning, obstacle avoidance of robot hand, and Cartesian space control strategies.

Trajectory planning involves interpolation and/or approximation of the desired Cartesian path by a class of polynomial functions of degree n or less. For the past several years, there has been a surge of efforts in the planning of manipulator trajectories which concern more sophisticated and efficient movements of the manipulator hand (or tool) along a straight line path or a joint-interpolated smooth trajectory which approximates the given path closely [8,33,34,37,42,46,47].

In general, trajectory planning in the joint-variable space is simpler than that in the Cartesian space because of problems caused by degeneracies and dynamic constraints [8]. The degenerate case happens when a set point on the manipulator hand trajectory in the Cartesian space results in a singular Jacobian matrix for the mappings between joint space and Cartesian space. The dynamic constraints are composed of maximum allowable velocity and acceleration of each joint and torque constraints depending on instantaneous joint position and velocity. In addition to the dynamic constraints, each joint trajectory must maintain a smooth transition which yields velocity and acceleration continuity from one set point to the next set point [8,46].

The trajectory planning of a manipulator path made up of straight line segments was discussed by Paul [48], using a homogeneous transformation matrix to represent target positions for the manipulator hand to traverse. Movement between two consecutive target positions is accomplished by two sequential operations: a

translation and a rotation to align the approach vector of the manipulator hand and a final rotation about the tool axis to align the gripper orientation. A quadratic polynomial interpolation routine in the joint-variable space is then used to guarantee smooth transition between two connected path segments.

Taylor [58] improved Paul's technique by using quaternions to represent the rotational operation. He also developed a bounded deviation joint control scheme which involved selecting more intermediate interpolation points when the joint polynomial approximation deviates too much from the desired straight line path. In order to achieve a real-time trajectory planning objective, both approaches neglect the physical manipulator dynamics.

Other existing trajectory planning schemes satisfy the continuity and the torque constraints simultaneously. To assure the torque constraint in the trajectory planning stage, it is assumed that the maximum allowable torque is known and independent of position and velocity. For example, instead of using a varying torque constraint, Lin et al. [34] use velocity, acceleration, and jerk[3] bounds which are assumed constant for each joint. They select several knot points on the desired Cartesian path, solve the inverse kinematics and obtain set points in joint space, and find cubic splines for the joint variables which guarantee continuity conditions and a fit through the set points in the joint-variable space. Then, by relaxing the normalized time to real time, the dynamic constraints on velocity, acceleration, and jerk are included.

However, due to the joint interpolation function, the exact location of the manipulator hand at every servo instant is not exactly on the desired path, but on a

[3]jerk is defined as the rate of change of acceleration.

related smooth function. This results in inaccurate straight line trajectory planning. Recently, Bobrow, et al. [7] and Shin, et al. [53] utilized a parameterized joint interpolation function to find the optimal trajectory on a given geometric path. Here we follow a similar approach but focus on very accurate implementation of a straight line trajectory which allows the detection of potential wrist collisions along the trajectory. This constitutes a part of the thesis and will be discussed in detail in Chapter 3.

Most of the existing off-line path planning schemes which concern obstacle avoidance concern the problem of avoiding fixed and stationary obstacles in a workspace. Ahuja, et al. [1] discussed two methods for detecting intersections of three dimensional objects that are based on the projection checking and the octree traversing of the stationary obstacles. Chien, et al.[11] adopted the concepts of state space and developed rotation mapping graph to describe the relationship between the positions and the corresponding collision-free orientations of a robot among obstacles.

To simplify the collision avoidance problem, Udupa [59] used a polyhedra model for the obstacles and minimum bounding cylinders in detecting the interference and avoiding collisions among three dimensional stationary objects. He discretized the three-dimensional space into sectoroids and pascs which were designated free if not occupied by obstacles and robots. A collision-free path is then obtained as lists of free pascs joined together. Lozano-Perez [35,36] and Brooks [9,10] extended the polyhedra models and obtained linear intersection functions for collision avoidance. Their schemes require special planning for the orientation problem.

Luh et al. [39-41] used the concept of pseudo-obstacles by expanding the real obstacles' edges and faces and letting the robot be viewed as a point location in space. They formulated the forbidden region for the Stanford arm as the union set of polygonal obstacles and pseudo-obstacles. An algorithm is developed to determine the shortest collision-free path for stationary obstacles given a sequence of edges to be traversed.

Pennington et al. [49] employed a geometric modeling approach to model a PUMA robot as a sphere model, where the collision is detected based on the swept volume of the robot. However, the collision detection was found quite involved in its mathematical computation. Petrov[50] proposed a learning algorithm to avoid the obstacles by considering the obstacles in joint space and in Cartesian space simultaneously, and showed a hybrid computer implementation of the algorithm for the collision avoidance.

Recently, Gilbert et al. [19] described a procedure where a collision-free optimal path for a robot is obtained by solving an optimal control problem where the distances between the potentially colliding parts of the robot and the obstacles impose a state space constraint. The state constraints and actuator constraints are handled numerically by a penalty function approach.

All the above collision avoidance schemes emphasize a single robot with a fixed environment of stationary obstacles and achieved various degrees of success. In this research, our concerns focus mainly on the wrist collision avoidance of the two moving robots in a common workspace.

Given the wrist collision-free path and trajectory, the purpose of manipulator control is to control the manipulator hand to track the desired trajectory as closely

as possible under various payload conditions. Most of the existing control schemes [5,13,21-23,25,26,44,57,65] control the arm at the joint level and emphasize nonlinear compensations of the interaction forces among the various joints. For most applications, resolved motion control, which commands the manipulator hand to move in a desired Cartesian direction in a coordinated position and rate control, may be more appropriate.

The resolved motion control algorithms [18,43,62-64] control the arm at the hand level with or without external sensory feedback information. Whitney developed a resolved motion rate control [62] in which motions of various joint motors are combined and resolved into separately controllable hand motions along the world coordinate axes. His method indicates that several joint motors must run simultaneously at different time-varying rates in order to achieve desired coordinated hand motion along the world coordinate axes. The method enables the user to specify the direction and speed along any arbitrarily oriented path. This motion control greatly simplifies the specification of the sequence of motion for completing a task because users are usually more adapted to the Cartesian coordinate system than to the joint coordinate system.

Luh et al. [43] extended the concept of resolved motion rate control to include acceleration control. It presents an alternative position control which deals directly with the position and orientation of the manipulator hand, and it assumes that the desired accelerations of a preplanned hand motion are specified by the user. The paper by Gilbert et al. [18] can be thought of as a generalization of this ideas.

The resolved motion control algorithms discussed may be inadequate for accurate path tracking control purposes because they neglect the changes of the load in

a task cycle. They may result in reduced servo response speed and damping, and limiting the precision and speed of the end-effector. As a result, the manipulator hand may deviate too much from the specified collision-free path, causing unexpected wrist collisions. A significant performance gain in this can be considered by adaptive control techniques in Cartesian space [28].

Recently, various adaptive control algorithms [3,13,21,23,25] have been proposed. Dubowsky et al. [13] proposed a model referenced adaptive control which uses a linear second-order time invariant differential equation as the referenced model for each degree of freedom of the robot arm. The manipulator is controlled by adjusting the position and velocity feedback gains to follow the model. A steepest descent method is used to update the feedback gains. Koivo et al. [23] proposed an adaptive self-tuning controller using an autoregressive model to fit the input-output data from the manipulator. Both control algorithms assume that the interaction forces among the joints are negligible. Lee et al. [25] proposed an adaptive perturbation control with feedforward compensation for robot manipulators in the joint-variable space.

A part of this research is devoted to extend the adaptive perturbation control to the Cartesian space so that the manipulator hand can be controlled to follow the desired collision-free trajectory faithfully in the Cartesian space.

1.4. Contribution and Organization of the Thesis

This thesis focuses on motion planning and motion control for two robots so that they can work safely in a common workspace. It involves the investigation of an off-line straight line trajectory planning scheme in Cartesian space, off-line collision-free path planning algorithms for two robots, and an on-line adaptive

perturbation control method in Cartesian space. Research results from these three areas constitute the major contribution of the thesis. They are now described briefly.

A discrete time trajectory planning scheme for a given straight line path has been developed to determine the trajectory set points which satisfy both smoothness and torque constraints. A real time servo interval instead of a normalized time interval is used for planning all the joint trajectories. The planning of the straight line trajectory is performed in the Cartesian space and requires maximization of the speed between two consecutive Cartesian set points on the straight line path. An iterative forward and backward search algorithm coupled with a modified forward search algorithm is developed to determine the joint values at each servo interval such that they are within the bounds imposed by the smoothness and torque constraints and satisfy the straight line requirement with high accuracy.

The resultant straight line trajectory is composed of three segments: an acceleration portion, a deceleration portion, and a relaxation portion. The control set points on the acceleration and deceleration portions of the trajectory satisfy the straight line requirement while those on the relaxation portion do not. These sets of joint position, velocity and acceleration can be used as reference inputs to the manipulator controller for the joint-variable space control [25,26] or can be used to determine the corresponding Cartesian position, velocity, and acceleration for the proposed Cartesian space adaptive control [28].

After obtaining the Cartesian trajectory set points from the trajectory planner, the potential wrist collisions between two robots can be detected easily by using a sphere model to represent the manipulator wrist. The detection of the potential

wrist collisions can be done by checking the distance between the center of the two spheres with respect to a global coordinate system. A collision-free path of a moving robot in the presence of a stationary robot is found from connected straight line segments selected subject to a performance measure. Various collision situations for two moving robots are identified. Notions of collision map, which provides the location and corresponding servo time information of the robots simultaneously, and time scheduling are introduced and utilized. Collision-free paths of two moving robots are obtained by considering the paths and trajectories of the two robots.

When the path modification is not allowed for collision avoidance, rescheduling the traveling time on the existing path is performed to realize collision-free motion planning. When the path modification is allowed for collision avoidance, a collision-free path is obtained subject to a performance measure.

An adaptive control in Cartesian space, which adopts the ideas of resolved motion rate control and resolved motion acceleration control, is proposed to control the manipulator hand to track the collision-free trajectory as closely as possible. Equations of motion of the manipulator hand are developed in Cartesian coordinates from which an adaptive control scheme is derived. The adaptive control scheme is based on the linearized perturbation equations along a desired hand trajectory.

The controlled system is characterized by feedforward and feedback components which can be computed separately and simultaneously. The feedforward component computes the nominal torques from the Newton-Euler equations of motion using the joint trajectory information from the trajectory planner. The feedback component, consisting of a recursive least square identification scheme and an optimal adaptive self-tuning controller for the linearized system, computes the

perturbation torques which reduce the manipulator hand position and velocity errors along the nominal hand trajectory. A feasibility study of implementing the adaptive control for a six-joint PUMA manipulator is explored using present-day low-cost microprocessors.

This thesis is organized in seven chapters. The problem of motion planning and motion control of two robots in a common workspace is formulated in three separate but coherent subproblems: trajectory planning, wrist modeling and collision-free path planning, and adaptive control in the Cartesian space. The hierarchical robot arm control structure is briefly discussed in Chapter 2 together with a discussion and problem formulation for each of the three areas of the proposed research.

In Chapter 3, the trajectory planning scheme in discrete time is presented for a straight line path. The objective of the trajectory planner is to obtain the control set points which are exactly on the specified straight line path and within the smoothness and torque constraints of the manipulator.

In Chapter 4, planning of a collision-free path is discussed for the two robot system. A collision-free path is found for a moving robot in the presence of a stationary robot, which consists of connected straight line segments selected subject to a performance measure. Various collision situations are identified and discussed for the case of two moving robots. Notions of collision map and time scheduling are introduced and two procedures are developed to obtain collision-free paths for the two moving robots.

In Chapter 5, an adaptive perturbation control is developed in Cartesian space to control the manipulator to track a collision-free trajectory faithfully. Equations

of motion of the manipulator hand are developed in Cartesian coordinates. A least square identification scheme together with an optimal adaptive self-tuning control is discussed. Computational complexity of the proposed adaptive perturbation control scheme is tabulated for future implementation.

In Chapter 6, computer simulation results of the proposed straight line trajectory planning scheme are presented. Sampling effects and the existence of a straight line trajectory are investigated. Also, the time scheduling of a straight line trajectory is performed and utilized in the collision map to obtain a collision-free trajectory for the case of two moving robots.

In Chapter 7, conclusions and summaries are presented. Extensions with future research issues are also discussed.

CHAPTER 2

PROBLEM FORMULATION

2.1. Introduction

This chapter addresses problem formulation for motion planning and motion control of a two robot system. The motion planning and motion control are investigated by focusing on three separate but coherent areas: planning of a straight line trajectory in Cartesian space, planning of a collision-free path of two robots, and manipulator control in Cartesian space.

The functional operation and characteristics of the hierarchical robot control structure are discussed in Section 2.2. Planning of a straight line trajectory in Cartesian space is discussed in Section 2.3. Several geometric models for the manipulator wrist used in planning a collision-free path are discussed with their advantages and disadvantages in Section 2.4. In Section 2.5, various collision situations are identified and discussed to obtain collision-free paths of the two robots. A Cartesian space control method is proposed in Section 2.6. Finally, summaries are presented in Section 2.7.

2.2. Hierarchical Robot Control Structure

In this section, the overall control structure of the robot system is discussed within the context of this thesis. In general, a task is performed in three phases of

execution: task planning, motion planning, and motion control of the manipulator. [55]

By the nature of the robot system, the overall control structure can be arranged hierarchically into three levels as shown in Figure 2.1. The highest system level performs task planning. The task planning system is composed of a human operator and a task planner. The human operator interacts with the task planner and initiates the task specification for the robot system to accomplish. The task planner decomposes the given task into sequences of operations for the robot to implement.

The intermediate system level consists of path planning and trajectory planning. The path planning system receives information about the locations of the various workpieces from the task planning system and generates the Cartesian path to be followed by the manipulator to accomplish the specified task. Then the trajectory planning system generates the time dependent control set points.

The lowest system level implements motion control. The motion control system receives control set points from the trajectory planning system and sensory feedback information to generate error actuating signals to servo the joint motors so that the manipulator will track the trajectory as closely as possible.

One way to control the two robots is to supervise and coordinate the motion of each robot by a supervisory computer [31]. Figure 2.2 shows the hierarchical structure of a two robot system with a supervisory computer. To control the two robots without collision in a common workspace, sharing and exchanging of sensory feedback information between the two robots must be performed in real time. A method of sharing and exchanging of sensory information is to design a real time

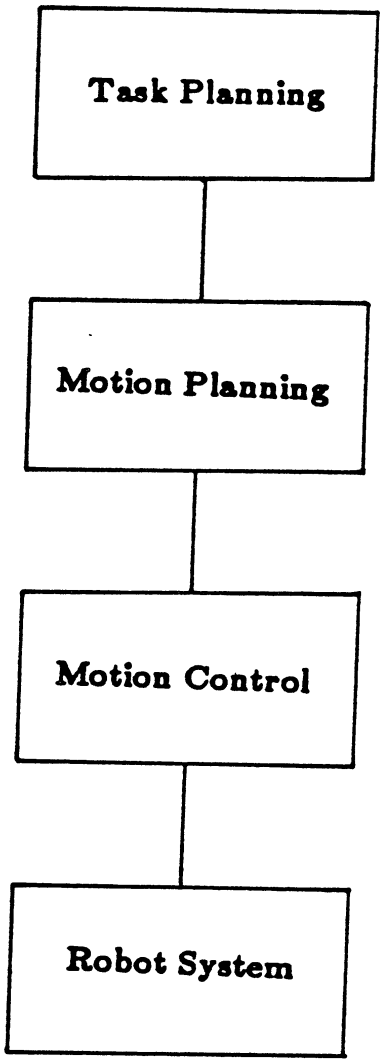


Figure 2.1 Hierarchical Control Structure of a Robot

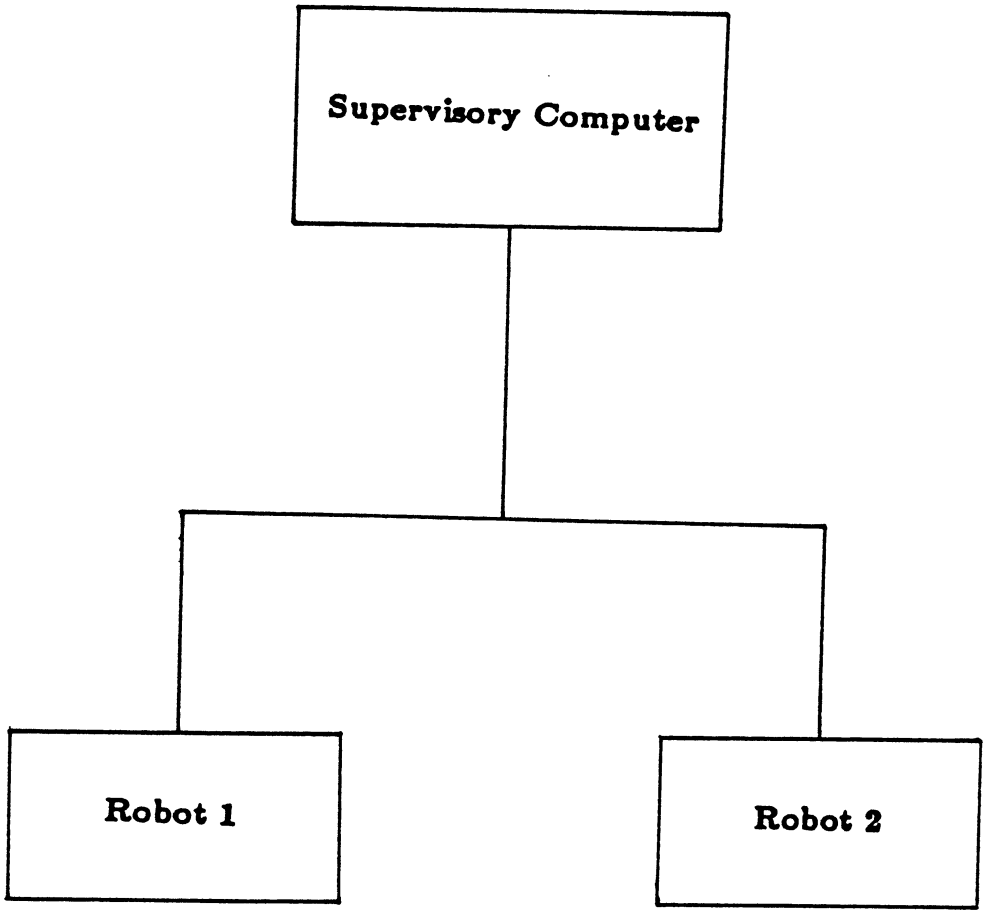


Figure 2.2 Two Cooperative Robot System

interactive controller for each robot using visual, force and touch sensory information as feedback signals. However, for a present-day robot system, many practical difficulties occur in designing such a sensor-based interactive controller.

As an approach to realizing appropriate interaction between the two robots, we plan the path and trajectory for both robots interactively. Due to the geometry of the wrist of a robot, potential wrist collisions along the pre-planned path may occur. Thus, a suitable geometric model of the wrist of the robot is needed. Furthermore, algorithms must be developed to replan the paths or trajectories if the potential wrist collisions occur along the paths.

2.3. A Scheme for Straight Line Trajectory Planning

Trajectory planning is part of motion planning, which converts the desired path information into a sequence of time-based intermediate configurations of the manipulator starting at the initial location and ending at the final location. Two currently available approaches to trajectory planning are briefly described. A scheme using discrete time approximation is proposed for straight line trajectory planning later.

The first approach requires the user to explicitly specify a set of constraints on position, velocity, and acceleration of the manipulator's joint coordinates at selected locations (called knot points) along the trajectory. Then a parameterized trajectory is selected from a certain class of functions (usually the class of piecewise polynomial functions of degree n or less, for some n, in the time interval $[\, t_0 , t_f \,]$) that "interpolates" and satisfies the constraints at the interpolation points. The second approach requires the user to explicitly specify the path that the manipulator hand has to traverse, such as a straight line path in Cartesian coordinates; and requires

the trajectory planner to determine a desired trajectory that closely "approximates" the desired path in Cartesian coordinates. These two approaches result in simple trajectories which are efficient, smooth, and accurate with a fast computation time (near real time).

For the purpose of controlling the manipulator, the joint trajectory function must be evaluated at each servo instant of time. In obtaining the control set points from the joint interpolation function, the resultant control set points are assumed to be within the torque constraints of the manipulator. However, due to the approximation of the specified path by the joint interpolation function, the control set points on the joint interpolation function do not satisfy exactly the specified path in the Cartesian space.

For example, a desired straight line path with several knot points A, B, C, $\cdots$ is shown in Figure 2.3. If we obtain an appropriate joint interpolation function through these knot points and evaluate the function for the control set points, the resultant control set points will be the points a_1, a_2, a_3, $\cdots$ in the joint-variable space, which corresponds to the points A_1, A_2, A_3, $\cdots$ in the Cartesian space, respectively. It is obvious that these points A_1, A_2, A_3, $\cdots$ may not be on the desired straight line path.

We now describe the relationships which exist between joint variables and Cartesian variables. These are necessary for us to describe our scheme of trajectory planning. The location of the manipulator hand with respect to a global coordinate system (x_0, y_0, z_0) can be realized by establishing an orthonormal coordinate frame at the manipulator hand which can be represented by a 4×4 hand coordinate transformation matrix [24,46],

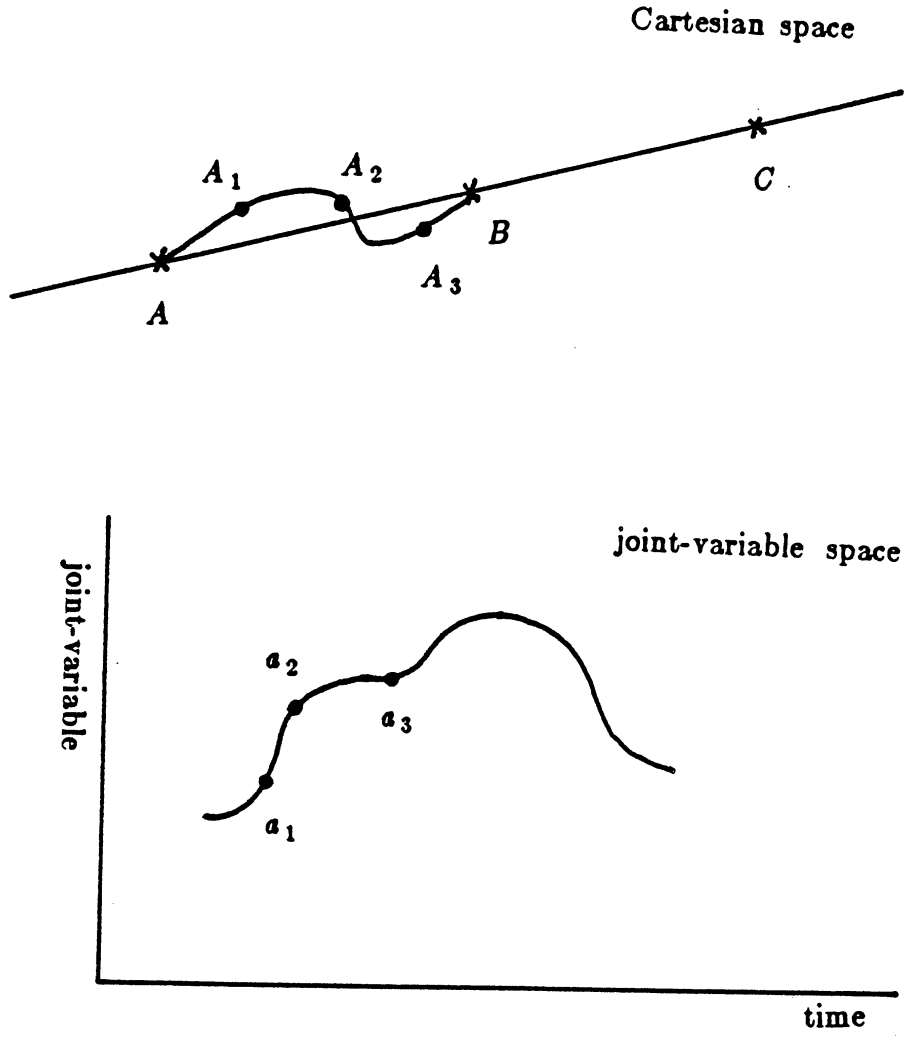


Figure 2.3 Joint Interpolated Trajectory

$$\mathbf{H}(t) = \begin{bmatrix} \mathbf{R}(t) & \mathbf{p}(t) \\ 0 & 1 \end{bmatrix} = \begin{bmatrix} \mathbf{n}(t) & \mathbf{s}(t) & \mathbf{a}(t) & \mathbf{p}(t) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

$$= \begin{bmatrix} n_x(t) & s_x(t) & a_x(t) & p_x(t) \\ n_y(t) & s_y(t) & a_y(t) & p_y(t) \\ n_z(t) & s_z(t) & a_z(t) & p_z(t) \\ 0 & 0 & 0 & 1 \end{bmatrix}, \tag{2.1}$$

where $\mathbf{p}(t)$ is the position vector of the manipulator hand; and $\mathbf{n}(t)$, $\mathbf{s}(t)$, $\mathbf{a}(t)$ are the unit normal vectors along the axes of the coordinate frame fixed to the manipulator hand. The dependence of the elements of the hand transformation matrix on joint variables is shown in the Appendix B.

Instead of using the rotation submatrix $[\mathbf{n}(t), \mathbf{s}(t), \mathbf{a}(t)]$ to describe the orientation, Euler angles (yaw ($\alpha(t)$), pitch ($\beta(t)$), and roll ($\gamma(t)$)) can be used to describe the rotation of the manipulator hand. The relationship between $[\mathbf{n}(t), \mathbf{s}(t), \mathbf{a}(t)]$ and $[\alpha(t), \beta(t), \gamma(t)]$ is:

$$\mathbf{R}(t) = \begin{bmatrix} n_x(t) & s_x(t) & a_x(t) \\ n_y(t) & s_y(t) & a_y(t) \\ n_z(t) & s_z(t) & a_z(t) \end{bmatrix} = \begin{bmatrix} C\gamma & -S\gamma & 0 \\ S\gamma & C\gamma & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} C\beta & 0 & S\beta \\ 0 & 1 & 0 \\ -S\beta & 0 & C\beta \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & C\alpha & -S\alpha \\ 0 & S\alpha & C\alpha \end{bmatrix} \tag{2.2}$$

$$= \begin{bmatrix} C\gamma C\beta & -S\gamma C\alpha + C\gamma S\beta S\alpha & S\gamma S\alpha + C\gamma S\beta C\alpha \\ S\gamma C\beta & C\gamma C\alpha + S\gamma S\beta S\alpha & -C\gamma S\alpha + S\gamma S\beta C\alpha \\ -S\beta & C\beta S\alpha & C\beta C\alpha \end{bmatrix}$$

where $\sin\alpha \equiv S\alpha$, $\cos\alpha \equiv C\alpha$, $\sin\beta \equiv S\beta$, $\cos\beta \equiv C\beta$, $\sin\gamma \equiv S\gamma$, $\cos\gamma \equiv C\gamma$.

In fact, $[\alpha(t), \beta(t), \gamma(t)]$ are the functions of joint variables as indicated in Appendices B and C. Define the position $\mathbf{p}(t)$, orientation $\Phi(t)$, linear velocity $\mathbf{v}(t)$, and angular velocity $\Omega(t)$ vectors of the manipulator hand with respect to the global coordinate frame, respectively, as:

$$\begin{aligned} \mathbf{p}(t) &\triangleq (\, p_x(t)\,,\, p_y(t)\,,\, p_z(t)\,)^T \quad ; \quad \Phi(t) \triangleq (\, \alpha(t)\,,\, \beta(t)\,,\, \gamma(t)\,)^T \quad ; \\ \mathbf{v}(t) &\triangleq (\, v_x(t)\,,\, v_y(t)\,,\, v_z(t)\,)^T \quad ; \quad \Omega(t) \triangleq (\, \omega_x(t)\,,\, \omega_y(t)\,,\, \omega_z(t)\,)^T . \end{aligned} \tag{2.3}$$

where the superscript T denotes the transpose operation. When we denote the joint position vector as $\mathbf{q}(t)$, the relationship between $(\, \mathbf{p}(t)\,,\, \Phi(t)\,)$ and $\mathbf{q}(t)$ can be represented as:

$$(\, \mathbf{p}^T(t)\,,\, \Phi^T(t)\,)^T = \mathbf{N}(\mathbf{q}(t)) \tag{2.4a}$$

or

$$\mathbf{q}(t) = \mathbf{N}^{-1}(\, \mathbf{p}(t)\,,\, \Phi(t)\,) \tag{2.4b}$$

where $\mathbf{N}(\cdot)$ is the function describing the kinematics routine and $\mathbf{N}^{-1}(\cdot)$ is the function describing the inverse kinematics routine.

The linear and angular velocity of the manipulator hand can be obtained from the lower joint velocities based on the moving coordinate frame concept [38],

$$\begin{bmatrix} \mathbf{v}(t) \\ \Omega(t) \end{bmatrix} = \Big[\mathbf{J}(\mathbf{q})\Big]\dot{\mathbf{q}}(t) = \Big[\, \mathbf{J}_1(\mathbf{q})\,,\, \mathbf{J}_2(\mathbf{q})\,,\, \cdots\,,\, \mathbf{J}_6(\mathbf{q}) \,\Big]\, \dot{\mathbf{q}}(t) \tag{2.5}$$

where $q_i\,(t) = \theta_i\,(t)$ if joint i is rotary, and $q_i\,(t) = d_i\,(t)$ if joint i is prismatic; $\dot{\mathbf{q}}(t) = (\dot{q}_1(t)\,,\, \cdots\,,\, \dot{q}_6(t))^T$ is the joint velocity vector of the robot; and $\mathbf{J}(\mathbf{q})$ is a 6×6 matrix whose i^{th} column vector $\mathbf{J}_i\,(\mathbf{q})$ can be found from

$$\mathbf{J}_i\,(\mathbf{q}) = \begin{cases} \begin{bmatrix} \mathbf{z}_{i-1} \times (\, \mathbf{p} - \mathbf{p}_{i-1}\,) \\ \mathbf{z}_{i-1} \end{bmatrix} & ; \text{ if } \textit{joint } i \textit{ is rotational} \\ \\ \begin{bmatrix} \mathbf{z}_{i-1} \\ 0 \end{bmatrix} & ; \text{ if } \textit{joint } i \textit{ is translational} \end{cases} \tag{2.6}$$

where $\times$ indicates vector cross product, $\mathbf{p}_{i-1}$ is the position of the origin of the $(i-1)^{th}$ coordinate frame with respect to the reference frame, $\mathbf{z}_{i-1}$ is the unit vector

along the axis of motion of joint i, and $\mathbf{p}$ is the position of the manipulator hand with respect to the reference coordinate frame.

The linear and angular acceleration of the manipulator hand can be obtained by taking the time derivative of the velocity vector in Eq. (2.5),

$$\begin{bmatrix} \dot{\mathbf{v}}(t) \\ \dot{\Omega}(t) \end{bmatrix} = [\dot{\mathbf{J}}(\mathbf{q}, \dot{\mathbf{q}})]\dot{\mathbf{q}}(t) + [\mathbf{J}(\mathbf{q})]\ddot{\mathbf{q}}(t) \tag{2.7}$$

where $\ddot{\mathbf{q}}(t) = (\ddot{q}_1(t), \cdots, \ddot{q}_6(t))^T$ is the joint acceleration vector of the robot.

Now, the equation of a straight line path is described. An initial location $(\mathbf{p}(t_0), \Phi(t_0))$ and a final location $(\mathbf{p}(t_f), \Phi(t_f))$ are given to the manipulator. The manipulator hand is required to move from the initial location to the desired final location along the straight line specified by these two end points. The straight line equation that passes through these two points can be described as:

$$\mathbf{p}(t) = \mathbf{p}(t_0) + \lambda(t) \cdot (\mathbf{p}(t_f) - \mathbf{p}(t_0)) \tag{2.8a}$$

where $0 \leq \lambda(t) \leq 1$. It is worth noting that t_f is not a fixed final traveling time but something we will find. Also, since $\Phi(t_0)$ and $\Phi(t_f)$ are specified, it is desirable that the manipulator hand reaches its final orientation angles when $\mathbf{p}(t)$ approaches $\mathbf{p}(t_f)$. This can be easily done by interpolating the Euler angles between $\Phi(t_0)$ and $\Phi(t_f)$ linearly along the given straight line:

$$\Phi(t) = \Phi(t_0) + \lambda(t) \cdot (\Phi(t_f) - \Phi(t_0)) \tag{2.8b}$$

where $\lambda(t)$ is the same as in Eq. (2.8a).

As previously discussed, currently available trajectory planning schemes are based on underlying joint interpolation function, which must be evaluated at every servo time instant for an input to the manipulator control system. For the control

system which we will consider, it is required that $[\, \mathbf{p}(t)\,,\ \Phi(t)\,,\ \mathbf{v}(t)\,,\ \Omega(t)\,,\ \dot{\mathbf{v}}(t)\,,\ \dot{\Omega}(t)\,]$ be known at every servo time instant. Since the detection of potential wrist collisions must be performed at each servo instant, it is desirable to have the control servo points (or set points) precisely on the straight line path given by Eqs. (2.8a) and (2.8b). To do this, a discrete time approach will be used.

In addition to the straight line requirement, the discretized control set points in the joint-variable space must be within certain limits to maintain and guarantee the smoothness of the trajectory. Also, due to the physical limitations of the joint motors, the required joint torques to move the robot from the current set point to the next set point must be within the torque constraint of the joint motors. Thus, the resultant control set points must satisfy the straight line requirement and must be within the smoothness and torque constraints of the robot so that these set points can be tracked by the manipulator hand faithfully.

A real time servo interval instead of a normalized time interval will be used in obtaining the control set points. Let us denote T as the servo time interval. Then we consider $\mathbf{p}(kT)$ when k is integer. Similarly, we consider $\Phi(kT)$ and $\lambda(kT)$. Ideally, we want to obtain the minimum time trajectory of the robot for the given path. There are many difficulties in obtaining the minimum time trajectory. Thus, instead we maximize the local traveling speed, which is identical to maximize $\lambda(kT) - \lambda(kT-T)$. More precisely, the trajectory planning problem is formulated as follows:

$$\textit{maximize} \qquad \| \lambda(kT) - \lambda(kT-T) \| \tag{2.9}$$

subject to the conditions that every $\mathbf{p}(kT)$ must be on the straight line and the

corresponding joint values from ($\mathbf{p}(kT)$, $\Phi(kT)$) must be within the smoothness and torque constraints. A detailed derivation of the straight line trajectory planning scheme is presented in Chapter 3.

2.4. Geometric Modeling of the Wrist of a Robot

The detection of potential wrist collisions cannot be handled easily due to their complex geometry. A simplified and appropriate geometric model of the wrist is necessary to detect any potential collisions along the preplanned trajectory [49]. We shall concentrate on investigating a suitable wrist model for a PUMA robot.

Figure 2.4 shows a cylindrical model and a truncated cone model for the wrist which is composed of the upper three links of the robot. For the detection of potential wrist collisions between two robots, all geometric models should be viewed with respect to the global coordinate frame $(\mathbf{x}_0,\mathbf{y}_0,\mathbf{z}_0)$. The mathematical expression of these models in the global coordinate frame become time dependent and very complicated, thus causing the detection of potential wrist collisions to be computationally intensive.

A sphere model, shown in Figure 2.5, is proposed in this thesis. The sphere contains links 4,5, and 6, including the workpiece grasped by the manipulator hand. The origin of the sphere, which coincides with the origin of the hand coordinate frame, is uniquely determined from displacement of the joints in the arm. The radius of the sphere is determined depending on the workpiece and the wrist geometry. The main advantage of this model is that the sphere is rotationally invariant. This reduces the complexity of the detection of potential wrist collisions to calculating the distance between the origins of the two spheres.

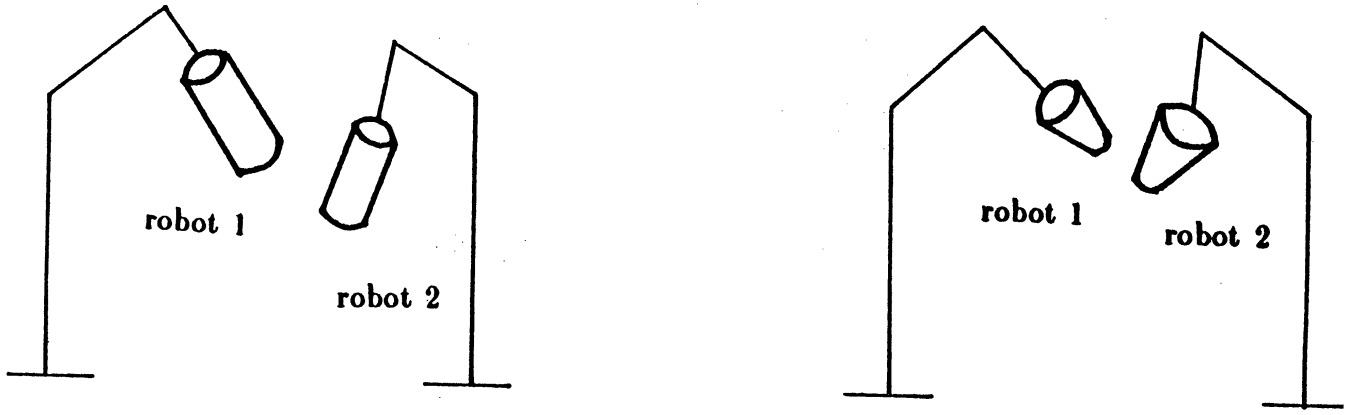


Figure 2.4 Cylindrical and Truncated Cone Model

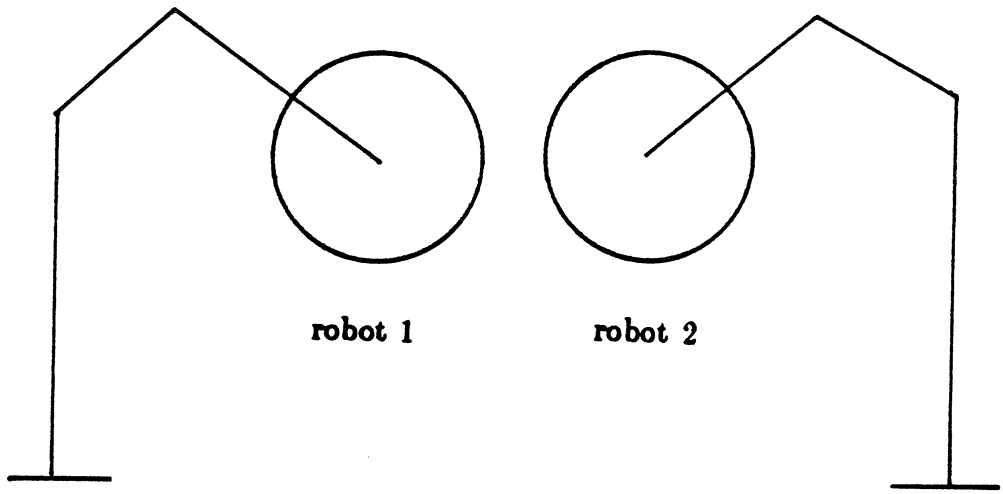


Figure 2.5 Sphere Model

In modeling the wrist geometry as a sphere, a large unnecessary space is included into the sphere so that computational efficiency in detecting the potential collisions is achieved at the expense of the modeling accuracy. The use of sphere model in obtaining a collision-free path is described in Chapter 4.

2.5. Various Collision Situations for Two Robot System

In this section, an investigation of various collision situations is presented. The terminology of "potential collision" is confined to the wrist collision between the two robots. The sphere model is used to model the wrists of each manipulator.

Two major classes of the potential collision are identified: (1) potential collision between a stationary robot and a moving robot; (2) potential collision between two moving robots. When two robots are working together in a common workspace, one robot may stop temporarily at one position, which corresponds to the class (1). On the other hand, two robots may work together in a common workspace, changing position and orientation simultaneously, which corresponds to the class (2).

Planning a collision-free path for class (1) is investigated in an assumed collision situation, where the sphere of a stationary robot wrist with radius r_1 (we label this robot as Robot 1) collides with the sphere of a moving robot wrist with radius r_2 (we label this robot as Robot 2) as shown in Figure 2.6. It is assumed that potential collisions occur between the two robots during the time interval from $t = mT$ to $t = nT$, where T is the servo sampling period for both robots. Various performance measures can be considered for obtaining a collision-free path in class (1). A collision-free path in this class will be found in Section 4.2.

Planning of a collision-free path for class (2) is investigated by considering the paths and trajectories of two robots simultaneously. It is important to note that

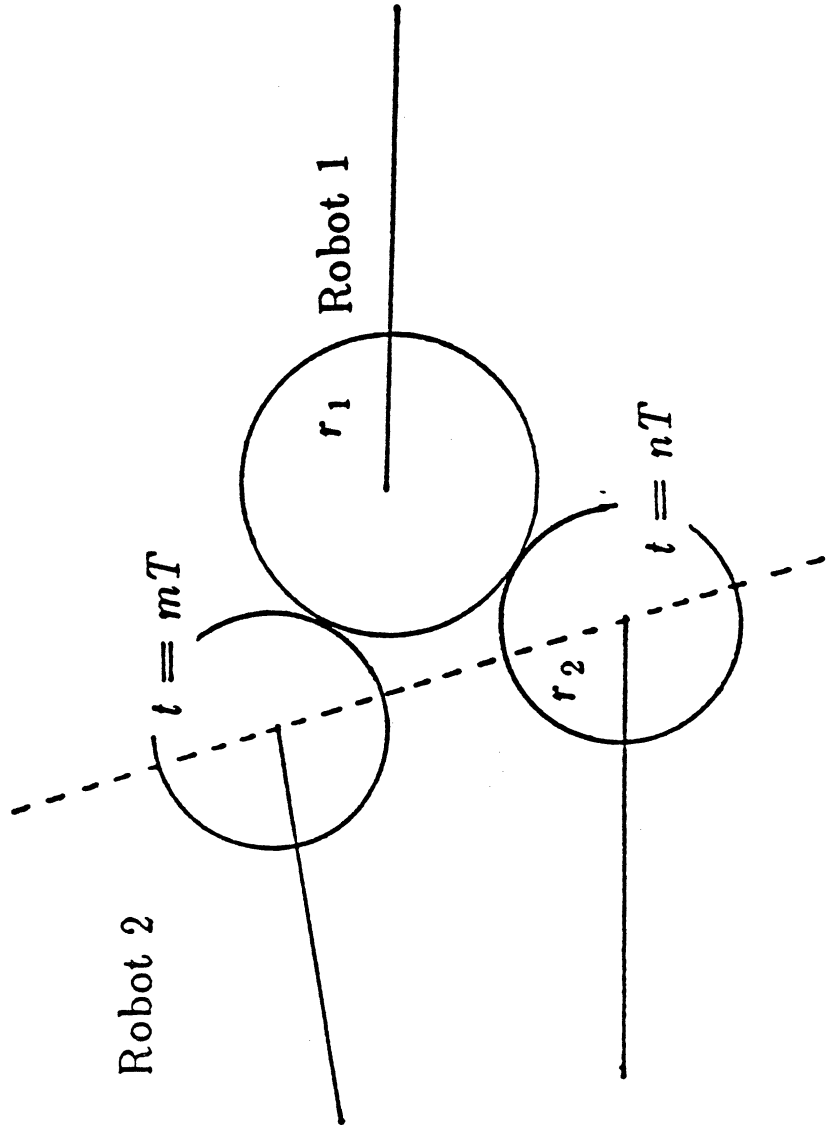


Figure 2.6 Two Colliding Robots

even if the original robot paths intersect, a potential collision may not occur because of the differences in the traveling time of the two robots along each path. Thus, the planning of a collision-free path in class (2) is viewed differently from that in class (1).

Assume that a straight line path and its trajectory information are given for robot 2 with the initial time t_0 and the final time t_f. Also, assume that a potential collision is detected and robot 2 has the lower priority such that it must change its path or trajectory information to avoid collision with robot 1. Here, we classify the collision situation into 4 cases on the assumption that the straight line trajectories of two robots are planned as discussed in Section 2.3.

Case 1: The final arrival time t_f of robot 2 can be relaxed but its original path cannot be changed because the modification of path may induce another collision with other obstacles.

Case 2: The final arrival time t_f of robot 2 can be relaxed and its original path can be changed.

Case 3: The final arrival time t_f of robot 2 cannot be relaxed but its original path can be changed.

Case 4: The final arrival time t_f of robot 2 and its original path cannot be relaxed.

In case 1, since only the final arrival time t_f of robot 2 can be changed, a method to obtain a different trajectory, which can eliminate the potential collision with robot 1, must be developed for the same path. We call this time scheduling of a straight line trajectory. In case 2, except for the initial and the final locations, the original path and trajectory information can be changed freely in order to avoid

potential collisions. There are a number of ways to obtain a collision-free path of robot 2. A collision-free path of robot 2 in this case will be found in Section 4.4.

In case 3, the final arrival time t_f is fixed while the original path can be changed. Due to the fixed final arrival time t_f, there is no guarantee that there exists a collision-free path of robot 2 satisfying the final arrival time t_f. In case 4, the original path and trajectory cannot be changed. That is, the collision avoidance must be realized by changing the path and/or trajectory of the other moving robot.

Only cases 1 and 2 are considered in this thesis; cases 3 and 4 are more difficult. Detailed derivations in planning a collision-free path are presented in Chapter 4.

2.6. A Cartesian Space Control

Most of the existing control schemes control the robot arm at the joint level and emphasize nonlinear compensation of the interaction forces among the various joints. One of the main reasons for performing control in the Cartesian space is that, in the joint-variable space control, a small joint error may cause a fairly large error in the Cartesian position of the manipulator hand. The error may be crucial in achieving the collision-free motion control for the two robots.

In general, it is easier to specify the desired motion of the robot in terms of a manipulator hand trajectory in Cartesian coordinates. However, the robot control system requires the reference inputs specified in joint coordinates. The mathematical relationship between these two coordinate systems is important in deriving a resolved motion control scheme. Hence, it is necessary to examine the kinematic relationships of position, velocity and acceleration between the joint and the Cartesian coordinate systems.

The location of the manipulator hand with respect to the global coordinate system can be realized by establishing an orthonormal hand coordinate frame at the gripping point of the manipulator hand as shown in Figure 2.7. The problem of finding the location of the manipulator hand is reduced to finding the position and orientation of the hand coordinate frame with respect to the global coordinate frame. This can be conveniently achieved by the 4×4 hand transformation matrix as in Eq. (2.1).

Using the same definition of linear and angular position, and velocity as in Eq. (2.3), the linear velocity of the manipulator hand with respect to the global coordinate frame is equal to the time derivative of the position of the manipulator hand,

$$\mathbf{v}(t) = \frac{d\,\mathbf{p}(t)}{dt} = \dot{\mathbf{p}}(t) \tag{2.10}$$

Since the inverse of a direction cosine matrix is equivalent to its transpose, the instantaneous angular velocity of the hand coordinate frame about the axes of the global coordinate frame can be obtained from Eq. (2.2) as [15],

$$\mathbf{R}\frac{d\mathbf{R}^T}{dt} = -\frac{d\mathbf{R}}{dt}\mathbf{R}^T = -\begin{bmatrix} 0 & -\omega_z & \omega_y \\ \omega_z & 0 & -\omega_x \\ -\omega_y & \omega_x & 0 \end{bmatrix} \tag{2.11}$$

$$= \begin{bmatrix} 0 & -S\,\beta\,\dot{\alpha} + \dot{\gamma} & -S\,\gamma\,C\,\beta\,\dot{\alpha} - C\,\gamma\,\dot{\beta} \\ S\,\beta\,\dot{\alpha} - \dot{\gamma} & 0 & C\,\gamma\,C\,\beta\,\dot{\alpha} - S\,\gamma\,\dot{\beta} \\ S\,\gamma\,C\,\beta\,\dot{\alpha} + C\,\gamma\,\dot{\beta} & -C\,\gamma\,C\,\beta\,\dot{\alpha} + S\,\gamma\,\dot{\beta} & 0 \end{bmatrix}$$

From Eq. (2.11), the relation between the $(\omega_x(t), \omega_y(t), \omega_z(t))^T$ and $(\dot{\alpha}(t), \dot{\beta}(t), \dot{\gamma}(t))^T$ can be found by equating the non-zero elements in the matrices,

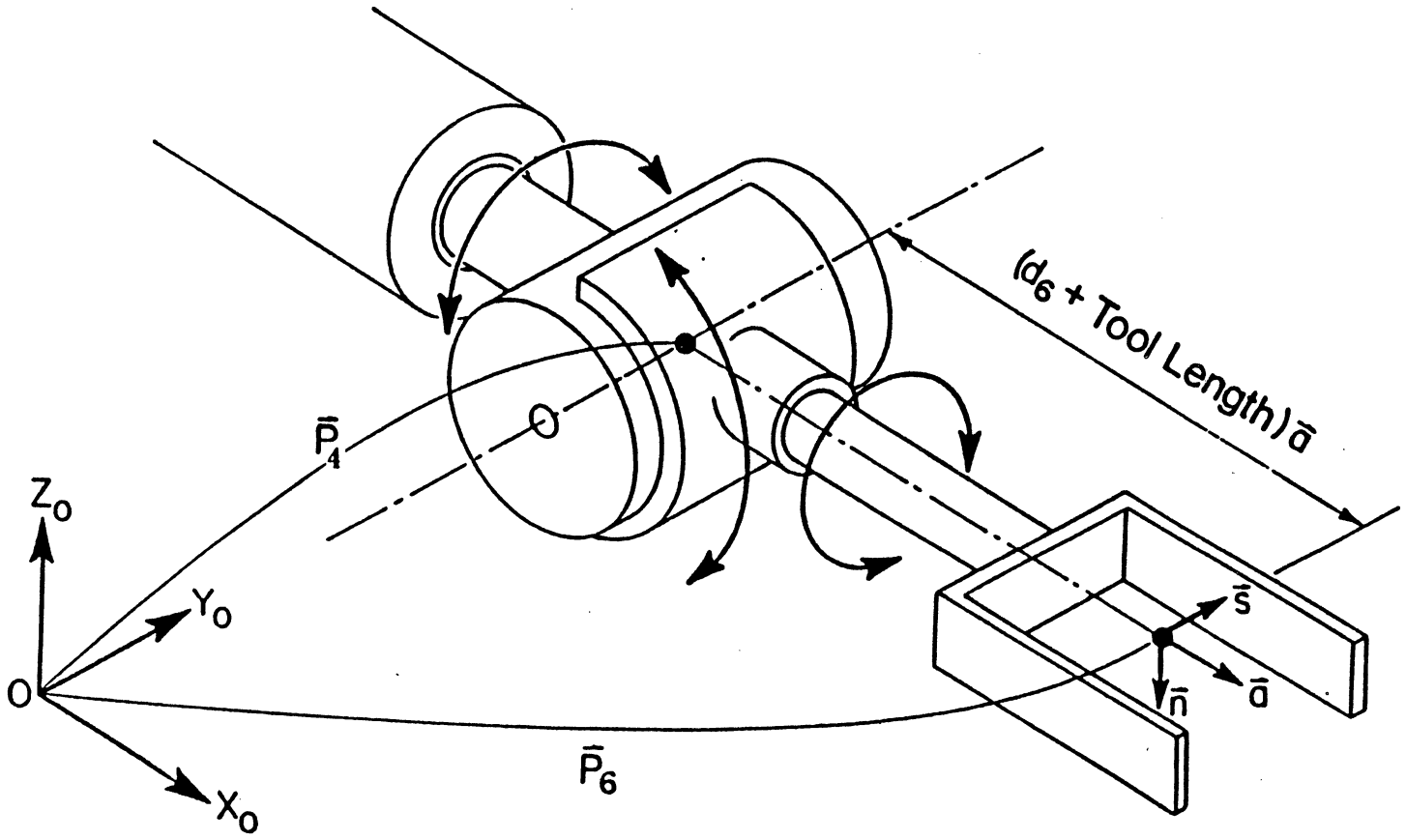


Figure 2.7 Hand Coordinate System

$$\begin{bmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{bmatrix} = \begin{bmatrix} C\gamma C\beta & -S\gamma & 0 \\ S\gamma C\beta & C\gamma & 0 \\ -S\beta & 0 & 1 \end{bmatrix} \begin{bmatrix} \dot{\alpha}(t) \\ \dot{\beta}(t) \\ \dot{\gamma}(t) \end{bmatrix} \tag{2.12}$$

Then, its inverse relation can be found easily,

$$\begin{bmatrix} \dot{\alpha}(t) \\ \dot{\beta}(t) \\ \dot{\gamma}(t) \end{bmatrix} = \sec\beta \begin{bmatrix} C\gamma & S\gamma & 0 \\ -S\gamma C\beta & C\gamma C\beta & 0 \\ C\gamma S\beta & S\gamma S\beta & C\beta \end{bmatrix} \begin{bmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{bmatrix}. \tag{2.13}$$

Using our vector notation, this becomes

$$\dot{\Phi}(t) \triangleq \left[\mathbf{M}(\Phi)\right] \Omega(t) \tag{2.14}$$

Based on the moving coordinate frame concept [38], the linear and angular velocities of the manipulator hand can be obtained from the velocities of the lower joints, as in Eq. (2.5).

If **J** is invertible at $\mathbf{q}(t)$, then the joint velocity $\dot{\mathbf{q}}(t)$ can be computed from the manipulator hand velocity using Eq. (2.5),

$$\dot{\mathbf{q}}(t) = [\,\mathbf{J}^{-1}(\mathbf{q})\,] \begin{bmatrix} \mathbf{v}(t) \\ \Omega(t) \end{bmatrix} \tag{2.15}$$

Given the desired linear and angular velocities of the manipulator hand, Eq. (2.15) computes the corresponding joint velocity and indicates the rates at which the joint motors must be maintained in order to achieve a steady hand motion along the desired Cartesian direction.

Using Eqs. (2.7) and (2.15), we have,

$$\begin{bmatrix} \dot{v}(t) \\ \dot{\Omega}(t) \end{bmatrix} = [\, \dot{J}(q\,,\,\dot{q})\,]\,[\, J^{-1}(q)\,]\,\begin{bmatrix} v(t) \\ \Omega(t) \end{bmatrix} + [\, J(q)\,]\,\ddot{q}(t) \qquad (2.16)$$

Then the joint acceleration $\ddot{q}(t)$ can be computed from the manipulator hand velocity and its derivative as,

$$\ddot{q}(t) = [\, J^{-1}(q)\,]\,\begin{bmatrix} \dot{v}(t) \\ \dot{\Omega}(t) \end{bmatrix} - [\, J^{-1}(q)\,]\,[\, \dot{J}(q\,,\,\dot{q})\,]\,[\, J^{-1}(q)\,]\,\begin{bmatrix} v(t) \\ \Omega(t) \end{bmatrix} \qquad (2.17)$$

The Eqs. (2.4b), (2.15), and (2.17) constitute the governing equations which transform the Cartesian coordinates to the joint coordinates.

Our objective is to utilize the resolved motion control concept to develop an adaptive control in the Cartesian space. Hence, it is necessary to develop an equations of motion of the manipulator in Cartesian coordinates. The above kinematic relations between the joint and Cartesian coordinates will be used to derive the equations of motion of the manipulator in Cartesian coordinates. Detailed derivation of the resolved motion adaptive control scheme is presented in Chapter 5.

2.7. Summary

The overall control structure of the robot system is arranged in three hierarchical levels: task planning, motion planning, and motion control. The servo-time-based trajectory planning scheme is proposed and the problems associated with the trajectory planning scheme are also discussed. Using a sphere model for the wrist of a robot, the detection of potential wrist collision is found to be computationally efficient at the expense of modeling accuracy. Various collision situations are identified and classified for obtaining the collision-free paths of two robots. Finally, an adaptive control in the Cartesian space which extends the resolved motion control concept has been suggested.

CHAPTER 3

PLANNING OF STRAIGHT LINE MANIPULATOR TRAJECTORY

3.1. Introduction

In this chapter, we develop a discrete time trajectory planning scheme to obtain the straight line trajectory set points which are applicable to avoiding collisions of the wrists between two moving robots, or between one stationary and one moving robot. Using the kinematic equations described in Section 2.3, the local speed optimization of the robot hand between two consecutive Cartesian set points is formulated along with a discussion of the various constraints in Section 3.2. Section 3.3 describes a binary search algorithm to find the servo control set points subject to the various constraints. The binary search technique generates servo control set points which result in an acceleration portion and a deceleration portion of the given straight line. Break point and relaxation points which are needed to construct the overall trajectories are discussed in Section 3.4. In Section 3.5, the trajectory planning of connected straight line segments is presented. Finally, summaries are given in Section 3.6.

3.2. Formulation for Local Speed Optimization

In this section, the optimization of local traveling speed along the given straight line is described subject to the straight line requirement, and smoothness

and torque constraints. It is based on the discretized joint velocity, acceleration, and jerk approximations.

Straight Line Constraint. In the straight line trajectory planning, the initial location [$\mathbf{p}(t_0)$, $\Phi(t_0)$] and the final location [$\mathbf{p}(t_{ft})$, $\Phi(t_f)$] are given for the manipulator, where $\mathbf{p}(t)$ and $\Phi(t)$ specify the Cartesian position and the Euler angles of the manipulator hand at time t, respectively. The manipulator hand is required to move from an initial location (position and orientation) to a desired final location in the Cartesian space along the straight line specified by these two end points. Using $\mathbf{p}(t) = (p_x(t), p_y(t), p_z(t))^T$, the straight line equation that passes through these two points is described by

$$\begin{aligned} p_x(t) &= \lambda(t) \cdot (p_x(t_{ft}) - p_x(t_0)) + p_x(t_0) \\ p_y(t) &= \lambda(t) \cdot (p_y(t_{ft}) - p_y(t_0)) + p_y(t_0) \\ p_z(t) &= \lambda(t) \cdot (p_z(t_{ft}) - p_z(t_0)) + p_z(t_0) \end{aligned} \qquad (3.1)$$

where $0 \leq \lambda(t) \leq 1$, and t_0 and t_{ft} are the initial and final time respectively. The time t_{ft} is not fixed, but will be determined for the robot by the overall trajectory planning algorithm. Thus we use the subscript ft to denote the final time to be tried or determined. Also, since $\Phi(t_0)$ and $\Phi(t_{ft})$ are specified, it is desirable that the manipulator hand reaches the final orientation angle when $\mathbf{p}(t)$ approaches $\mathbf{p}(t_{ft})$. Using $\Phi(t) = (\alpha(t), \beta(t), \gamma(t))^T$, this can be achieved by interpolating the Euler angles between $\Phi(t_0)$ and $\Phi(t_{ft})$ linearly using the ratio of the traveled length to the total length of the given straight line as described in the following equation,

$$\begin{aligned}
\alpha(t) &= \lambda(t)\cdot(\,\alpha(t_{ft}) - \alpha(t_0)\,) + \alpha(t_0)\\
\beta(t) &= \lambda(t)\cdot(\,\beta(t_{ft}) - \beta(t_0)\,) + \beta(t_0)\\
\gamma(t) &= \lambda(t)\cdot(\,\gamma(t_{ft}) - \gamma(t_0)\,) + \gamma(t_0)
\end{aligned} \tag{3.2}$$

The position $\mathbf{p}(t)$ and the Euler angles $\Phi(t)$ can be augmented into a 6×1 vector and described by:

$$\begin{bmatrix}\mathbf{p}(t)\\ \Phi(t)\end{bmatrix} = \mathbf{N}(\mathbf{q}(t)) = (N_1(\mathbf{q}(t)), \ldots, N_6(\mathbf{q}(t)))^T \tag{3.3}$$

where $N(.)$ is a 6×1 nonlinear vector function depending on the manipulator configuration.

Currently available trajectory planning schemes are based on underlying joint interpolation functions. The joint interpolation function must be evaluated at every servo time instant and the joint values must be stored in the computer memory as an input to the manipulator control system. For our control scheme, it is required that $[\,\mathbf{p}(t)\,,\,\Phi(t)\,,\,\mathbf{v}(t)\,,\,\Omega(t)\,,\,\dot{\mathbf{v}}(t)\,,\,\dot{\Omega}(t)\,]$ be known at every instant of sampling period for the Cartesian space control [28] and that these control set points be precisely on a straight line. The algorithm which we present uses a discrete time approach to find these data directly at every sampling instant.

To initiate the discretized trajectory analysis, let us denote the sampling period for the servo control of the robot as T (usually $3\ ms \leq T \leq 20\ ms$) and $\bar{\mathbf{q}}(k)$ to represent angular displacement $\mathbf{q}(kT)$:

$$\mathbf{q}(kT) \cong \bar{\mathbf{q}}(k)\ ;\ k = 0, 1, \cdots \tag{3.4}$$

Then we will approximate the velocity, the acceleration, and the jerk, respectively, at time kT by:

$$\dot{q}(kT) \cong \frac{1}{T}(\bar{q}(k) - \bar{q}(k-1)) = \dot{\bar{q}}(k) \tag{3.5}$$

$$\ddot{q}(kT) \cong \frac{1}{T^2}(\bar{q}(k) - 2\,\bar{q}(k-1) + \bar{q}(k-2)) = \ddot{\bar{q}}(k) \tag{3.6}$$

$$w(kT) \cong \frac{1}{T^3}(\bar{q}(k) - 3\,\bar{q}(k-1) + 3\,\bar{q}(k-2) - \bar{q}(k-3)) = \bar{w}(k) \tag{3.7}$$

where $w(kT)$ denotes the jerk at time $t = kT$. For simplicity and brevity, we shall abuse notation and drop the super bar from the rest of the equations. That is,

$$q(k) \equiv \bar{q}(k) \tag{3.8a}$$

$$\dot{q}(k) \equiv \dot{\bar{q}}(k) \tag{3.8b}$$

$$\ddot{q}(k) \equiv \ddot{\bar{q}}(k) \tag{3.8c}$$

$$w(k) \equiv \bar{w}(k) \tag{3.8d}$$

Finally, we let $t_0 = k_0 T$, where k_0 is assumed to be zero, and $t_{fi} = k_{fi} T$.

Smoothness Constraint. Since we will obtain the control set points not from a joint interpolation polynomial function but from the discrete time approximation, all discretized control set points in the joint variable space must be within certain limits to maintain and guarantee the smoothness of the trajectory. The smoothness constraint on the joint trajectory set points can be stipulated in three possible ways by a velocity bound (VB), an acceleration bound (AB), and a jerk bound (JB). They are given respectively as:

$$| \dot{q}_i(k) | \leq \epsilon_i^v \;;\; \epsilon_i^v > 0 \text{ and } i = 1, \cdots, 6 \tag{3.9}$$

$$| \ddot{q}_i(k) | \leq \epsilon_i^a \;;\; \epsilon_i^a > 0 \text{ and } i = 1, \cdots, 6 \tag{3.10}$$

$$| w_i(k) | \leq \epsilon_i^J \;;\; \epsilon_i^J > 0 \text{ and } i = 1, \cdots, 6 \tag{3.11}$$

where ϵ_i^v, ϵ_i^a, and ϵ_i^J are the i^{th} element of 6-dimensional bound vectors for the

manipulator. The velocity bound (VB) and acceleration bound (AB) constrain the joint actuators from exceeding the maximum limits of the velocity and acceleration. The jerk bound (JB) reduces wear of joint actuators, reduces excitation of vibrations, and constrains the smoothness. Hence, we impose VB, AB, and JB on the entire trajectory from one trajectory set point to another. Combining Eqs. (3.10) and (3.11), we have:

$$\ddot{q}_{i,min}(k) \leq \ddot{q}_i(k) \leq \ddot{q}_{i,max}(k) \quad ; \; i = 1, 2, \cdots, 6 \tag{3.12}$$

where

$$\ddot{q}_{i,min}(k) = \max \left\{ -\epsilon_i^a, \quad \ddot{q}_i(k-1) - \epsilon_i^J \cdot T \right\} \tag{3.13}$$

$$\ddot{q}_{i,max}(k) = \min \left\{ \epsilon_i^a, \quad \ddot{q}_i(k-1) + \epsilon_i^J \cdot T \right\} \tag{3.14}$$

Similarly, we have the following equation from Eqs. (3.9) and (3.12) for the joint velocity constraint.

$$\dot{q}_{i,min}(k) \leq \dot{q}_i(k) \leq \dot{q}_{i,max}(k) \quad ; \; i = 1, 2, \cdots, 6 \tag{3.15}$$

where

$$\dot{q}_{i,min}(k) = \max \left\{ -\epsilon_i^v, \quad T \cdot \ddot{q}_{i,min}(k) + \dot{q}_i(k-1) \right\} \tag{3.16}$$

$$\dot{q}_{i,max}(k) = \min \left\{ \epsilon_i^v, \quad T \cdot \ddot{q}_{i,max}(k) + \dot{q}_i(k-1) \right\} \tag{3.17}$$

Then, we can obtain the joint position constraint as:

$$q_{i,min}(k) \leq q_i(k) \leq q_{i,max}(k) \quad ; \; i = 1, 2, \cdots, 6 \tag{3.18}$$

where

$$q_{i,min}(k) = T \cdot \dot{q}_{i,min}(k) + q_i(k-1)$$
$$q_{i,max}(k) = T \cdot \dot{q}_{i,max}(k) + q_i(k-1) \tag{3.19}$$

Torque Constraint. Due to the physical limitations of the joint motors, the robot is limited on how far it can move from current set point to the next set point on the straight line path. The required joint torques to move the robot are described by its dynamic equations of motion and depend on the instantaneous joint position, velocity, acceleration, and the load that it is carrying. In general, the dynamical behavior of a robot can be described by the Lagrange-Euler equations of motion as [24,30,46]

$$\tau(kT) \cong \overline{\tau}(k) = [\,\mathbf{D}(\mathbf{q}(k))\,]\ddot{\mathbf{q}}(k) + \mathbf{h}(\mathbf{q}(k), \dot{\mathbf{q}}(k)) + \mathbf{c}(\mathbf{q}(k)) \tag{3.20}$$

where $\overline{\tau}(k)$ is an 6×1 applied torque vector for joint motors, $\mathbf{c}(\mathbf{q}(k))$ is an 6×1 gravitational force vector, $\mathbf{h}(\mathbf{q}(k), \dot{\mathbf{q}}(k))$ is an 6×1 Coriolis and centrifugal force vector, and $[\,\mathbf{D}(\mathbf{q}(k))\,]$ is an 6×6 acceleration-related matrix. The approximate equality results from the discrete-time approximation of $\mathbf{q}$, $\dot{\mathbf{q}}$ and $\ddot{\mathbf{q}}$. Hereafter we omit the super bar from $\overline{\tau}(k)$.

If $\mathbf{q}(k)$, $\dot{\mathbf{q}}(k)$ and $\ddot{\mathbf{q}}(k)$ are given, the required piecewise joint torques can be computed by treating the equations of motion as an inverse dynamics problem. In a simplified notation,

$$\tau(k) = [\,\mathbf{D}_k\,]\ddot{\mathbf{q}}(k) + \mathbf{h}_k + \mathbf{c}_k \tag{3.21}$$

where $\mathbf{D}_k = \mathbf{D}(\mathbf{q}(k))$, $\mathbf{h}_k = \mathbf{h}(\mathbf{q}(k), \dot{\mathbf{q}}(k))$ and $\mathbf{c}_k = \mathbf{c}(\mathbf{q}(k))$. At $k_0 = 0$, we assume:

$$\dot{\mathbf{q}}(0) = 0 \quad ; \quad \ddot{\mathbf{q}}(0) = 0 \quad ; \quad \mathbf{w}(0) = 0 \tag{3.22}$$

Let us further assume that the torques generated from Eq. (3.21) are constrained by

limits that are dependent on the joint position (due to the manipulator actuator geometry) and on the joint velocity (due to the back electromotive force terms or other actuator effects) as,

$$\tau_{i,min}\,(\mathbf{q}(k)\,,\,\dot{\mathbf{q}}(k)) \;\leq\; \tau_i\,(k) \;\leq\; \tau_{i,max}\,(\mathbf{q}(k)\,,\,\dot{\mathbf{q}}(k)) \;\;;\;\; i = 1,\;\cdots\;,6 \tag{3.23}$$

or in a simplified notation as,

$$\tau_{i,min}\,(k) \;\leq\; \tau_i\,(k) \;\leq\; \tau_{i,max}\,(k) \quad ;\; i = 1,\;\cdots\;,6 \tag{3.24}$$

Since the joint torque is represented by Eq. (3.21), the Eq. (3.24) can be written as,

$$\tau_{i,a}^{-}(k) \leq \mathbf{D}_{i,k}\,\ddot{\mathbf{q}}(k) \leq \tau_{i,a}^{+}(k) \tag{3.25}$$

where $\mathbf{D}_{i,k}$ represents the i^{th} row of the matrix $\mathbf{D}_k$, and $\tau_{i,a}^{-}(k)$ and $\tau_{i,a}^{+}(k)$ are written as:

$$\tau_{i,a}^{-}(k) = \tau_{i,min}\,(k) - \mathbf{h}_{i,k} - \mathbf{c}_{i,k} \tag{3.26}$$

$$\tau_{i,a}^{+}(k) = \tau_{i,max}\,(k) - \mathbf{h}_{i,k} - \mathbf{c}_{i,k} \tag{3.27}$$

where $\mathbf{h}_{i,k}$ and $\mathbf{c}_{i,k}$ are the i^{th} element of the vectors $\mathbf{h}_k$ and $\mathbf{c}_k$, respectively. Therefore, the joint values at each servo time instant must satisfy the straight line requirement given by Eqs. (3.1)-(3.2), the smoothness constraint given by Eq. (3.18), and the torque constraint given by Eq. (3.25).

Formulation for Optimization. The Eqs. (3.1)-(3.2) for the straight line requirement, Eq. (3.18) for the smoothness constraints, and Eq. (3.25) for the torque constraints will be used as constraints for finding the set points on the straight line path in an optimization process. We define

$$\bar{\mathbf{p}}(k) \cong \mathbf{p}(kT) = (\; p_x(kT),\, p_y(kT),\, p_z(kT)\;)^T \tag{3.28}$$

$$\bar{\Phi}(k) \cong \Phi(kT) = (\; \alpha(kT),\, \beta(kT),\, \gamma(kT)\;)^T \tag{3.29}$$

Again we drop the super bar from $\bar{\mathbf{p}}(k)$ and $\bar{\Phi}(k)$ for notational consistency. Let us assume that at the time $t = (k-1)T$, $\mathbf{p}(k-1)$ and $\Phi(k-1)$ are given and that they are within the physical bounds from the straight line requirements, the smoothness and torque constraints at $t = (k-1)T$. Then, we would like to find the next set point, $[\mathbf{p}(k), \Phi(k)]$, such that it is again within the smoothness and torque constraints and must lie on the specified straight line path exactly.

It is desirable to obtain the minimum time trajectory of the straight line path. However, there are many arising difficulties in obtaining the minimum time trajectory of the straight line path due to the discrete time approximation. Here, we would like to minimize the local traveling time during one servo time period, which does not yield the minimum time trajectory for the given straight line path. We maximize the Cartesian distance between two consecutive set points

$$|| \mathbf{p}(k) - \mathbf{p}(k-1) ||, \tag{3.30}$$

subject to the smoothness constraint, torque constraint, and the straight line requirements of $\mathbf{p}(k)$. We introduce $\lambda(k)$ which satisfies the following equation,

$$\begin{bmatrix} \mathbf{p}(k) \\ \Phi(k) \end{bmatrix} = \lambda(k) \cdot \begin{bmatrix} \mathbf{p}(k_{ft}) - \mathbf{p}(k_0) \\ \Phi(k_{ft}) - \Phi(k_0) \end{bmatrix} + \begin{bmatrix} \mathbf{p}(k_0) \\ \Phi(k_0) \end{bmatrix} \tag{3.31}$$

where $0 \leq \lambda(k) \leq 1$. From Eq. (3.3),

$$\begin{bmatrix} \Delta\mathbf{p}(k) \\ \Delta\Phi(k) \end{bmatrix} \simeq [\nabla \mathbf{N}(\mathbf{q}(k))] \cdot \Delta\mathbf{q}(k) \tag{3.32}$$

where $\Delta\mathbf{p}(k) = \mathbf{p}(k) - \mathbf{p}(k-1)$, $\Delta\Phi(k) = \Phi(k) - \Phi(k-1)$, $\Delta\mathbf{q}(k) = \mathbf{q}(k) - \mathbf{q}(k-1)$, and the elements of $[\nabla \mathbf{N}(\mathbf{q}(k))]$ are found to be

$$[\nabla \mathbf{N}(\mathbf{q}(k))]_{ij} = \frac{\partial \mathbf{N}_i(\mathbf{q}(k))}{\partial q_j(k)} \quad ; \; i,j = 1, 2, \cdots, 6 \tag{3.33}$$

Utilizing Eq. (3.31) at time $t = (k-1)T$ and $t = kT$ and their difference yields,

$$\begin{bmatrix} \Delta \mathbf{p}(k) \\ \Delta \Phi(k) \end{bmatrix} = \Delta\lambda(k) \begin{bmatrix} \mathbf{p}(k_{ft}) - \mathbf{p}(k_0) \\ \Phi(k_f) - \Phi(k_0) \end{bmatrix} \tag{3.34}$$

where $\Delta\lambda(k) = \lambda(k) - \lambda(k-1)$. Combining Eqs. (3.32) and (3.34), we have:

$$[\nabla \mathbf{N}(\mathbf{q}(k))] \cdot \Delta \mathbf{q}(k) \cong \Delta\lambda(k) \begin{bmatrix} \mathbf{p}(k_{ft}) - \mathbf{p}(k_0) \\ \Phi(k_{ft}) - \Phi(k_0) \end{bmatrix} \tag{3.35}$$

If $[\nabla \mathbf{N}(\mathbf{q}(k))]$ is non-singular at time $t = kT$, then

$$\Delta \mathbf{q}(k) \cong \Delta\lambda(k)[\nabla \mathbf{N}(\mathbf{q}(k))]^{-1} \begin{bmatrix} \mathbf{p}(k_{ft}) - \mathbf{p}(k_0) \\ \Phi(k_{ft}) - \Phi(k_0) \end{bmatrix} \tag{3.36}$$

We can rewrite Eq. (3.36) as:

$$\Delta \mathbf{q}(k) \cong \Delta\lambda(k)\mathbf{Q}(k) \tag{3.37}$$

where

$$\begin{aligned} \mathbf{Q}(k) &= [\nabla \mathbf{N}(\mathbf{q}(k))]^{-1} \begin{bmatrix} \mathbf{p}(t_{ft}) - \mathbf{p}(t_0) \\ \Phi(t_{ft}) - \Phi(t_0) \end{bmatrix} \\ &= [Q_1(k), \cdots, Q_6(k)]^T \end{aligned} \tag{3.38}$$

Physically, $\mathbf{Q}(k)$ is the vector which relates the angular displacement of each joint with $\Delta\lambda(k)$ of a given straight line. Since the servo time interval T is very small, let us assume that, for the joint position at $t = kT$,

$$\mathbf{q}(k) = \mathbf{q}(k-1) + \Delta \mathbf{q}(k) \tag{3.39}$$

Then using Eq. (3.37), we have,

$$\mathbf{q}(k) \cong \mathbf{q}(k-1) + \Delta\lambda(k)\mathbf{Q}(k) \tag{3.40}$$

Combining Eqs. (3.18), (3.19) and (3.40), we have:

$$\frac{T \cdot q_{i,min}(k)}{Q_i(k)} \leq \Delta\lambda(k) \leq \frac{T \cdot \dot{q}_{i,max}(k)}{Q_i(k)} \quad ; \; i = 1, 2, \cdots, 6 \tag{3.41}$$

Another constraint on $\Delta\lambda(k)$ is included from the torque constraint given by Eq. (3.25). Combining Eqs. (3.25) and (3.40), we have:

$$\Delta\lambda_{i,min}(k) \leq \Delta\lambda(k) \leq \Delta\lambda_{i,max}(k) \; ; \; i = 1, \cdots, 6 \tag{3.42}$$

where

$$\begin{aligned} \Delta\lambda_{i,min}(k) &= \frac{T^2\tau_{i,s}^{-}(k) + \mathbf{D}_{i,k}\,\mathbf{q}(k-1) - \mathbf{D}_{i,k}\,\mathbf{q}(k-2)}{\mathbf{D}_{i,k}\,\mathbf{Q}(k)} \\ \Delta\lambda_{i,max}(k) &= \frac{T^2\tau_{i,s}^{+}(k) + \mathbf{D}_{i,k}\,\mathbf{q}(k-1) - \mathbf{D}_{i,k}\,\mathbf{q}(k-2)}{\mathbf{D}_{i,k}\,\mathbf{Q}(k)} \end{aligned} \tag{3.43}$$

$\Delta\lambda_{i,min}(k)$ and $\Delta\lambda_{i,max}(k)$ are the maximum and minimum constraints for $\Delta\lambda(k)$ from the i^{th} joint torque constraint..

Since the maximum $\Delta\lambda(k)$ results in the maximum local speed at $t = kT$, it is desirable to formulate the problem as:

maximize $\Delta\lambda(k)$; k = 1, 2, ...

subject to the smoothness constraint

$$\frac{T \cdot q_{i,min}(k)}{Q_i(k)} \leq \Delta\lambda(k) \leq \frac{T \cdot q_{i,max}(k)}{Q_i(k)} \tag{3.44}$$

and the torque constraint

$$\Delta\lambda_{i,min}(k) \leq \Delta\lambda(k) \leq \Delta\lambda_{i,max}(k) \tag{3.45}$$

where $i = 1,2, \cdots, 6$. If we rewrite equation (3.44) as

$$\Delta\lambda_i^{-}(k) \leq \Delta\lambda(k) \leq \Delta\lambda_i^{+}(k) \tag{3.46}$$

where

$$\Delta\lambda_i^+(k) = \frac{T \quad q_{i,max}(k)}{Q_i(k)}$$
$$\Delta\lambda_i^-(k) = \frac{T \quad q_{i,min}(k)}{Q_i(k)} \tag{3.47}$$

for $i = 1, \cdots, 6$, then the problem can be formulated as follows:

maximize $\Delta\lambda(k)$; k = 1, 2, ...

subject to

$$\Delta\lambda_{i,min}(k) \leq \Delta\lambda(k) \leq \Delta\lambda_{i,max}(k)$$

and

$$\Delta\lambda_i^-(k) \leq \Delta\lambda(k) \leq \Delta\lambda_i^+(k)$$

where $i = 1, \cdots, 6$.

Since the constraints on $\Delta\lambda(k)$ may not be consistent, the existence of the maximum of $\Delta\lambda(k)$ is not always guaranteed. The existence of the straight line trajectory is discussed in Section 3.4. In fact, $\Delta\lambda_i^+(k)$, $\Delta\lambda_i^-(k)$, $\Delta\lambda_{i,min}(k)$ and $\Delta\lambda_{i,max}(k)$ are obtained and derived by approximations. Hence, it is very difficult to satisfy inequalities (3.45) and (3.46) exactly. Computational aspects of (3.45) and (3.46) are considered in the following discussion.

Computational Aspects The computations of $\Delta\lambda_i^+(k)$ and $\Delta\lambda_i^-(k)$ require the computations of $[\nabla N(q(k))]$ and its inverse, if it exists. Also, the computations of $\Delta\lambda_{i,min}$ and $\Delta\lambda_{i,max}$ require the actual torque constraints in Eq. (3.23) which are dependent on $q(k)$ and $\dot{q}(k)$. The elements of the nonlinear vector function $N(q(k)) = (N_1(q(k)), \cdots, N_6(q(k)))^T$ for the PUMA robot (Appendix C) are as follows:

$$N_1(q(k)) = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2) \tag{3.48}$$

$$N_2(q(k)) = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_3C_2] + C_2(d_6S_4S_5 + d_2) \tag{3.49}$$

$$N_3(q(k)) = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2 \tag{3.50}$$

$$N_4(q(k)) = \tan^{-1}\left[\frac{a_x(k)\sin(\tan^{-1}\frac{n_y(k)}{n_x(k)}) + a_y(k)\cos(\tan^{-1}\frac{n_y(k)}{n_x(k)})}{-s_x(k)\sin(\tan^{-1}\frac{n_y(k)}{n_x(k)}) + s_y(k)\cos(\tan^{-1}\frac{n_y(k)}{n_x(k)})}\right] \tag{3.51}$$

$$N_5(q(k)) = \tan^{-1}\left[\frac{-n_z(k)}{n_x(k)\cos(\tan^{-1}\frac{n_y(k)}{n_x(k)}) + \sin(\tan^{-1}\frac{n_y(k)}{n_x(k)})}\right] \tag{3.52}$$

$$N_6(q(k)) = \tan^{-1}\left[\frac{n_y(k)}{n_x(k)}\right] \tag{3.53}$$

where

$$n_x(k) = C_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] - S_2(S_4C_5C_6 + C_4C_6) \tag{3.54}$$

$$n_y(k) = S_1[C_{23}(C_4C_5C_6 - S_4S_6) - S_{23}S_5C_6] + C_2(S_4C_5C_6 + C_4C_6) \tag{3.55}$$

$$n_z(k) = -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6 \tag{3.56}$$

$$a_x(k) = C_2(C_{23}C_4S_5 + S_{23}C_5) - S_2S_4S_5$$

$$a_y(k) = S_1(C_{23}C_4S_5 + S_{23}C_5) + C_2S_4S_5 \tag{3.57}$$

$$s_x(k) = C_1[-C_{23}(C_4C_5C_6 + S_4S_6) + S_{23}S_5S_6] - S_1(-S_4C_5S_6 + C_4C_6) \tag{3.58}$$

and

$$s_y(k) = S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + C_2(-S_4C_5S_6 + C_4C_6) \tag{3.59}$$

where $C_i = \cos q_i$, $S_i = \sin q_i$, $C_{ij} = \cos(q_i + q_j)$ and $S_i = \sin(q_i + q_j)$

The computations of $[\nabla \mathrm{N}(\mathrm{q}(k))]_{ij}$, $i, j = 1, \cdots, 6$, require intensive mathematical operations especially for the orientation part of $\mathrm{N}(\mathrm{q}(k))$, that is, $N_4(\mathrm{q}(k))$,$N_5(\mathrm{q}(k))$, and $N_6(\mathrm{q}(k))$. In addition, the computation of arc tangent values may generate errors due to the table look-up method or the power series expansion in actual computations. The inverse of $[\nabla \mathrm{N}(\mathrm{q}(k))]$ will also require a moderate computational effort. In addition to the mathematical complexity, all the computation involved is based on a first order approximation, thus it is very difficult to obtain the exact constraint on $\Delta\lambda(k)$. The inequalities (3.46) from the torque constraints are also based on an approximation.

Due to the computational difficulties and the errors from the first order approximation, we resort to a search algorithm to find the maximum of $\Delta\lambda(k)$. Let $\mathrm{p}(k)$ be the reachable hand position of the robot from $\mathrm{p}(k-1)$ and $\Phi(k)$ be the corresponding Euler angles at time $t = kT$. If the inverse kinematics solution of $[\mathrm{p}(k), \Phi(k)]$ is within the smoothness and torque constraints, then finding the maximum of $\Delta\lambda(k)$ corresponds to finding the maximum of $\| \mathrm{p}(k) - \mathrm{p}(k-1) \|$.

We define l_k as:

$$l_k \overset{\Delta}{=} \| \mathrm{p}(k) - \mathrm{p}(k-1) \| = \Delta\lambda(k) \cdot \| \mathrm{p}(k_{ft}) - \mathrm{p}(k_0) \| \tag{3.60}$$

In the next section, we utilize a search algorithm to get the approximated maximum

value of $\Delta\lambda(k)$.

3.3. Search Algorithm

The method to get the approximated maximum value of $\Delta\lambda(k)$ involves three stages of development and utilizes a binary search technique to determine the largest value of path length ratio $\Delta\lambda(k)$ at time $t = kT$ within all the physical constraints.

Let us use a "hat" notation as $\hat{f}$ to denote the estimated value of the corresponding function f. Then, three stages of development are as follows. First, using the directional information of the straight line, $\hat{\lambda}(1)$ is determined. Recall that $\hat{\lambda}(1) = \Delta\hat{\lambda}(1)$. The initial estimate, $\Delta\hat{\lambda}^1(1)$, is then used in an iterative binary search algorithm to obtain the largest value of $\Delta\lambda(1)$. Here, we use the superscript 1 to denote the initial estimated value of the $\Delta\hat{\lambda}(1)$. Second, using $\Delta\lambda(k-1)$ and the projection of the estimate of linear velocity $\hat{\mathbf{v}}(k)$ on to the straight line, the initial estimate of $\Delta\hat{\lambda}^1(k)$, for $k = 2, 3, \cdots$, can be obtained. Third, the initial estimate $\Delta\hat{\lambda}^1(k)$ is used in the same iterative binary search algorithm to find the corresponding largest value of $\Delta\lambda(k)$ for $k = 2, 3, \cdots$.

From Figure 3.1, $\overline{AB}$ indicates the straight line path that the robot has to travel. At point A, $\mathbf{p}(k_0)$, $\Phi(k_0)$, $\mathbf{v}(k_0)$, $\Omega(k_0)$ are specified and satisfy the all constraints. We would like to find the control set points $\mathbf{p}(k)$, $\Phi(k)$, $\mathbf{v}(k)$, $\Omega(k)$, $\dot{\mathbf{v}}(k)$, $\dot{\Omega}(k)$ on the straight line together with the corresponding $\mathbf{q}(k)$, $\dot{\mathbf{q}}(k)$ and $\ddot{\mathbf{q}}(k)$ for $k = 1, \cdots$ such that the robot hand will traverse the straight line. Passing the $(\mathbf{p}(k_0), \Phi(k_0))$ values into the inverse kinematics solution routine, the corresponding joint angles, $\mathbf{q}(k_0) = \mathbf{N}^{-1}(\mathbf{p}(k_0), \Phi(k_0))$ are obtained. We assume, without loss of generality,

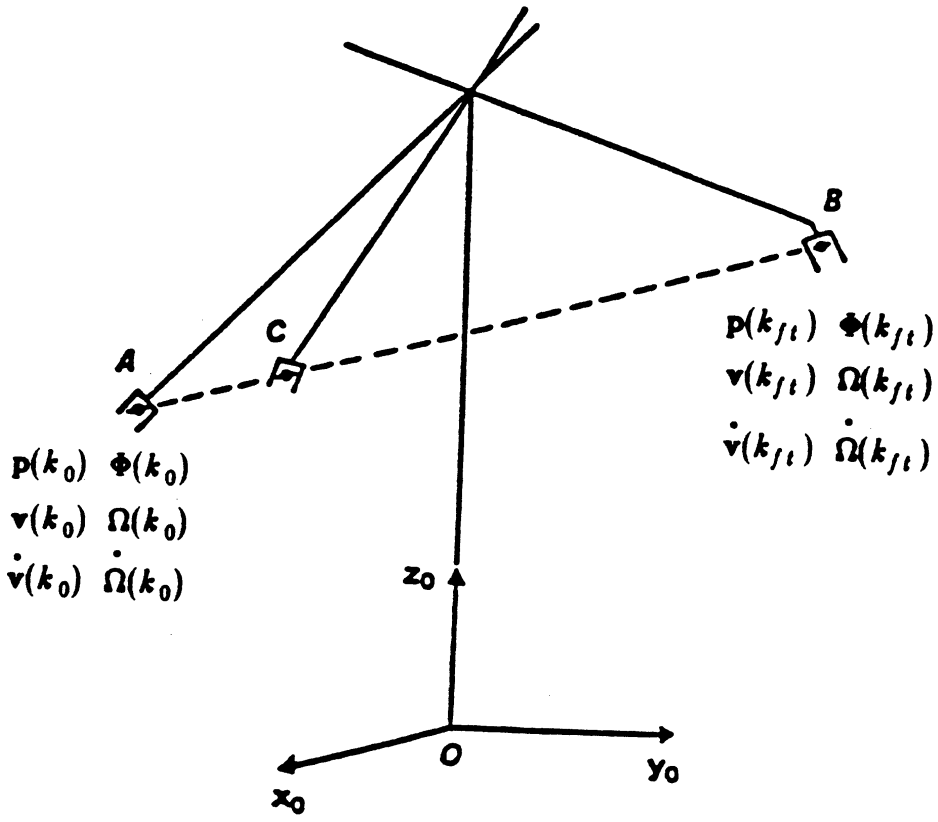


Figure 3.1 Straight Line Path

that $\mathbf{v}(k_0) = \Omega(k_0) = \dot{\mathbf{v}}(k_0) = \dot{\Omega}(k_0) = 0$ which lead to $\dot{\mathbf{q}}(k_0) = \ddot{\mathbf{q}}(k_0) = 0$. We further assume that $\mathbf{w}(k_0) = 0$. With these assumptions, a method to obtain $\Delta\hat{\lambda}(k)$ is discussed in the next section.

3.3.1. Method to Find Initial Estimated Length Ratio

$\lambda(1)$ can be estimated by evaluating the directional information of the given straight line. From the joint position constraint at time $t = T$ (Eq. (3.18)), $q_i(1)$ must be bounded by $q_{i,min}(1)$ and $q_{i,max}(1)$. Depending on the straight line equation, the linear velocity of the robot hand, $\mathbf{v}(1)$, must be constrained.

Let us denote the ij^{th} element of the Jacobian matrix which is defined in Eq. (2.6) as J_{ij}. From Eq. (2.5),

$$\begin{aligned}
v_x(1) &= J_{11}(1)\dot{q}_1(1) + \cdots + J_{16}(1)\dot{q}_6(1) \\
v_y(1) &= J_{21}(1)\dot{q}_1(1) + \cdots + J_{26}(1)\dot{q}_6(1) \\
v_z(1) &= J_{31}(1)\dot{q}_1(1) + \cdots + J_{36}(1)\dot{q}_6(1) \\
\omega_x(1) &= J_{41}(1)\dot{q}_1(1) + \cdots + J_{46}(1)\dot{q}_6(1) \\
\omega_y(1) &= J_{51}(1)\dot{q}_1(1) + \cdots + J_{56}(1)\dot{q}_6(1) \\
\omega_z(1) &= J_{61}(1)\dot{q}_1(1) + \cdots + J_{66}(1)\dot{q}_6(1)
\end{aligned} \tag{3.61}$$

Since the robot motion is expected to be very slow at the first servo time instant, we assume that $J_{ij}(1) \approx J_{ij}(0)$. If $v_x(1) > 0$ from the given straight line, then $\dot{q}_j(1)$ can be selected positively whenever $J_{1j}(1) > 0$, otherwise it can be selected negatively. That is, if $v_x(1) > 0$, then

$$\hat{\dot{q}}_j(1) = \begin{cases} q_{j,max}(1) & \text{if } J_{1j}(0) \geq 0 \\ q_{j,min}(1) & \text{if } J_{1j}(0) < 0 \end{cases} \quad ; \; j = 1, 2, \cdots, 6 \tag{3.62}$$

Similarly, if $v_x(1) < 0$, then

$$\hat{q}_j(1) = \begin{cases} q_{j,max}(1) & \text{if } J_{1j}(0) < 0 \\ q_{j,min}(1) & \text{if } J_{1j}(0) \geq 0 \end{cases} \quad ; \; j = 1, 2, \cdots, 6 \tag{3.63}$$

From the chosen values of $\hat{q}_j$ (1), we can obtain $\hat{p}_x(1)$ from the kinematic equations. Let us use a notation $\hat{\lambda}_x(1)$ to represent the estimated value of $\lambda(1)$ from the x directional requirement. Using Eq. (3.1), we can determine $\hat{\lambda}_x(1)$ from the directional constraint along the x-axis of the reference frame,

$$\hat{\lambda}_x(1) = \frac{\hat{p}_x(1) - p_x(k_0)}{p_x(k_{ft}) - p_x(k_0)} \tag{3.64a}$$

where if $p_x(k_{ft}) = p_x(k_0)$, then there is no constraint on $\hat{\lambda}_x(1)$. Similarly, along the y axis and z axis of the reference frame, we have, respectively,

$$\hat{\lambda}_y(1) = \frac{\hat{p}_y(1) - p_y(k_0)}{p_y(k_{ft}) - p_y(k_0)} \tag{3.64b}$$

$$\hat{\lambda}_z(1) = \frac{\hat{p}_z(1) - p_z(k_0)}{p_z(k_{ft}) - p_z(k_0)} \tag{3.64c}$$

where if $p_y(k_{ft}) = p_y(k_0)$, then there is no constraint on $\hat{\lambda}_y(1)$, and if $p_z(k_{ft}) = p_z(k_0)$, then there is no constraint on $\hat{\lambda}_z(1)$. Similarly, the signs of ω_x, ω_y and ω_z can be determined from the given Euler angle information and Eq. (2.12). That is, if $\omega_x(1) > 0$, then

$$\hat{q}_j(1) = \begin{cases} q_{j,max}(1) & \text{if } J_{4j}(0) \geq 0 \\ q_{j,min}(1) & \text{if } J_{4j}(0) < 0 \end{cases} \quad ; \; j = 1, 2, \cdots, 6 \tag{3.65}$$

and if $\omega_x(1) < 0$, then

$$\hat{q}_j(1) = \begin{cases} q_{j,max}(1) & \text{if } J_{4j}(0) < 0 \\ q_{j,min}(1) & \text{if } J_{4j}(0) \geq 0 \end{cases} \quad ; \; j = 1, 2, \cdots, 6 \tag{3.66}$$

Thus we have another set of constraints on $\hat{\lambda}(1)$ from the Euler angle information.

That is, we have $\hat{\lambda}_\alpha(1)$, $\hat{\lambda}_\beta(1)$ and $\hat{\lambda}_\gamma(1)$ as in Eqs. (3.64a)-(3.64c). Then we can choose $\hat{\lambda}(1)$ as:

$$\hat{\lambda}(1) = \min \{ \hat{\lambda}_x(1), \hat{\lambda}_y(1), \hat{\lambda}_z(1), \hat{\lambda}_\alpha(1), \hat{\lambda}_\beta(1), \hat{\lambda}_\gamma(1) \} \tag{3.67}$$

Recall that $\Delta\hat{\lambda}(1) = \hat{\lambda}(1)$ by the definition of $\lambda(k)$. Physically the choice of the value of $\hat{\lambda}(1)$ as in Eq. (3.67) means that the robot motion at $t = T$ is constrained critically by the corresponding position or Euler angle information.

3.3.2. The Ratio of Initial Estimated Path Length

Once $\hat{\lambda}(1)$ is found from Eq. (3.67), it can be used in Eqs. (3.1)-(3.2) to find $\hat{\mathbf{p}}(1)$ and $\hat{\Phi}(1)$. It should be noted that the $[\hat{\mathbf{p}}(1), \hat{\Phi}(1)]$ is on the given straight line exactly. However, the inverse kinematics solution from $\hat{\mathbf{p}}(1)$ and $\hat{\Phi}(1)$ may not be within the smoothness and torque constraints due to the fact that the value $\Delta\hat{\lambda}^1(1)$ is determined by one directional constraint in Eq. (3.67). In order to find the largest value of $\Delta\lambda(1)$ within the smoothness and torque constraints, the $\Delta\hat{\lambda}^1(1)$ is used in the steps FW4 to FW8 in the iterative binary forward search algorithm (Algorithm FW) in the next section. Once $\Delta\lambda(1)$ is found, we need to determine the initial estimate of $\Delta\hat{\lambda}^1(k)$, for $k = 2, 3, \cdots$ on the straight line and the corresponding largest value of $\Delta\lambda(k)$ from the same binary search algorithm.

Since we utilize the binary search algorithm to find $\Delta\lambda(k)$, it is reasonable to have an initial estimated value of $\Delta\hat{\lambda}^1(k)$ satisfying the smoothness and torque constraints to guarantee the existence of the solution. The torque constraint can be verified in the iterative search algorithm by checking whether the computed torques satisfy the torque constraints (Eq. (3.25)) or not. For the smoothness constraint, the best initial estimate of $\hat{q}_i(k)$, which corresponds to $\Delta\hat{\lambda}^1(k)$, is chosen to be the

mid-point of the allowable position bounds as:

$$\hat{q}_i(k) = \frac{1}{2}\,(q_{i,min}(k) + q_{i,max}(k)) \;\; ; \;\; i = 1, 2, ..., 6 \tag{3.68}$$

Using $\hat{q}_i(k)$ and $q_i(k-1)$, the initial estimate of joint velocity is found as:

$$\hat{\dot{q}}_i(k) \approx \frac{1}{T}(\hat{q}_i(k) - q_i(k-1)) \quad ; \; i = 1, 2, ..., 6 \tag{3.69}$$

The initial estimate of linear velocity, $\hat{\mathbf{v}}(k)$, can be obtained by using Eq. (2.5) as:

$$\hat{\mathbf{v}}(k) = \mathbf{J}_u(\hat{\mathbf{q}}(k))\,\hat{\dot{\mathbf{q}}}(k) \tag{3.70}$$

where $\mathbf{J}_u(\hat{\mathbf{q}}(k))$ is the upper 3×6 submatrix of the Jacobian matrix in Eq. (2.6). To obtain the estimated value of $\Delta\hat{\lambda}^1(k)$, let $\hat{v}_p(k)$ be the projection of $\hat{\mathbf{v}}(k)$ onto the given straight line specified by the initial position $\mathbf{p}(k_0)$ and the final position $\mathbf{p}(k_{ft})$. Then

$$\hat{v}_p(k) = \frac{(\,\mathbf{p}(k_{ft}) - \mathbf{p}(k_0)\,) \cdot \hat{\mathbf{v}}(k)}{\|\,\mathbf{p}(k_{ft}) - \mathbf{p}(k_0)\,\|} \tag{3.71}$$

Let us denote the total length of the given straight line path as l_{total}. Then, the projection of the linear velocity $\hat{\mathbf{v}}(k)$ onto the given straight line approximately corresponds to the value of $\dfrac{\Delta\hat{\lambda}(k)\cdot l_{total}}{T}$ at $t = kT$ because the linear velocity is obtained at the point on the straight line path and the projection of the velocity is just a component of the velocity in the straight line direction. That is,

$$\hat{v}_p(k)\cdot T \approx \Delta\hat{\lambda}(k)\cdot l_{total} \tag{3.72}$$

Thus, we use the following approximation at $t = kT$.

$$\Delta\hat{\lambda}^1(k) \approx \frac{\hat{v}_p(k)\cdot T}{l_{total}} \tag{3.73}$$

which leads to the initial estimate of $\Delta\hat{\lambda}^1(k)$ as,

$$\Delta\hat{\lambda}^1(k) = T \cdot \frac{(\mathbf{p}(k_{ft}) - \mathbf{p}(k_0)) \cdot \hat{\mathbf{v}}(k)}{\| \mathbf{p}(k_{ft}) - \mathbf{p}(k_0) \| \cdot l_{total}} \quad ; \; k = 2, 3, \cdots \tag{3.74}$$

The estimate $\Delta\hat{\lambda}^1(k)$ will be used to obtain the value of $\Delta\lambda(k)$ from which the Cartesian position and the Euler angles can be obtained and converted into the control set point at $t = kT$ on the straight line path. However, we have to mention that even if it is rational to take the initial estimate as in Eq. (3.74), the inverse kinematics solution from $\Delta\hat{\lambda}^1(k)$ may not always satisfy the smoothness and torque constraints.

3.3.3. Forward Search Algorithm

After finding the initial estimate of $\Delta\hat{\lambda}^1(k)$, our objective is to find the largest value of $\Delta\hat{\lambda}(k)$ on the straight line subject to the smoothness and torque constraints. This is solved by an iterative forward binary search algorithm which is used to determine all the control set points for the "acceleration" portion of the given straight line. The algorithm FW is outlined below.

Algorithm FW: Given the initial estimates of $\Delta\hat{\lambda}^1(k)$, this algorithm performs an iterative forward search of the largest value of $\Delta\lambda(k)$ on the given straight line.

FW1. [Initialize and loop] For each $k = 1, 2, \cdots$, perform steps FW2 to FW8.

FW2. [Compute the desired set of constraint limits] Compute the constraint limits for each joint:
(1) Smoothness constraint as in Eq. (3.18): $q_{i,min}(k)$, $q_{i,max}(k)$
(2) Torque constraint as in Eq. (3.25): $\tau_{i,a}^{-}(k)$, $\tau_{i,a}^{+}(k)$

FW3. [Initial estimate] Find $\Delta\hat{\lambda}^1(k)$ (Eq. (3.67) and (3.74)) and set $m = 0$. Also, set $\Delta\lambda^{low}(k) = \rho_1 \cdot \Delta\hat{\lambda}^1(k)$ and $\Delta\lambda^{high}(k) = \rho_2 \cdot \Delta\hat{\lambda}^1(k)$, where ρ_1 and ρ_2 are designated constants.
(when $k = m = 1$, $\Delta\lambda^{low}(1) = 0$)

[redacted]

then the bisection process between $\Delta\lambda^{low}(k)$ and $\Delta\lambda^{high}(k)$ will continue until $m = 14$. Simulation results with the failure of the search algorithm will be shown in Chapter 6.

3.4. Determination of Break Point and Relaxation Points

3.4.1. The Break Point and The Modified Search Algorithm

The forward search algorithm (Algorithm FW) generates the trajectory set points which constitute the acceleration portion of the straight line. Thus, a point must be determined on the straight line so that the deceleration portion starts from the point to meet the boundary conditions at the final location. A break point is defined as a point on the given straight line at which the acceleration portion of the straight line ends and the deceleration portion of the straight line starts. (Later, we will expand the definition of the break point to include a transition point from a deceleration portion to an acceleration portion also.)

Based on the initial position/orientation and linear/angular velocity information, [$\mathbf{p}(k_0)$, $\Phi(k_0)$, $\mathbf{v}(k_0)$, $\Omega(k_0)$], the forward search algorithm generates the largest values of $\Delta\lambda(1)$, $\Delta\lambda(2)$, $\cdots$, $\Delta\lambda(s)$, $\cdots$ which in turn generate the corresponding trajectory set points, $[\mathbf{p}(1)$, $\Phi(1)]$, $[\mathbf{p}(2)$, $\Phi(2)]$, $\cdots$, $[\mathbf{p}(s)$, $\Phi(s)]$, $\cdots$, on the straight line and at the same time the inverse kinematics solution of these locations satisfy the smoothness and torque constraints. Similarly, based on the final position/orientation and linear/angular velocity information, [$\mathbf{p}(k_{ft})$, $\Phi(k_{ft})$, $\mathbf{v}(k_{ft})$, $\Omega(k_{ft})$], the backward search algorithm (Algorithm BW), which is identical to the algorithm FW but starts from the final location, generates the largest values of

$\Delta\lambda_b(1)$, $\Delta\lambda_b(2)$, $\cdots$, $\Delta\lambda_b(r)$, $\cdots$ which corresponds to the trajectory set points, $[p_b(1), \Phi_b(1)]$, $[p_b(2), \Phi_b(2)]$, $\cdots$, $[p_b(r), \Phi_b(r)]$, $\cdots$, satisfying the smoothness and torque constraints also. We use the subscript b in $\Delta\lambda_b(k)$, $p_b(k)$ and $\Phi_b(k)$, where $k = 1, 2, \cdots$, to clarify that they are obtained from the backward search algorithm.

In generating the backward trajectory set points, we really do not know the time k_{fb}, but we can re-index the time from zero. In our estimation of the break point, we must then translate the backward trajectory set points in time so as to obtain a good fit of the forward and backward trajectories. The forward and backward binary search algorithms provide us the information about the position of a break point. The backward search algorithm generates the set points which do not constitute the control set points in the correct traveling direction of the robot. Also, the jerk bound may be violated at the break point if the backward search points are used from the break point. Thus, a modified forward search algorithm (Algorithm MFW) is used to find the set points from the break point to the final location.

This break point can be determined from the following procedure BR:

Procedure BR: Given the forward trajectory set points $[p(1), \Phi(1)]$, $[p(2), \Phi(2)]$, $\cdots$, $[p(s), \Phi(s)]$, $\cdots$ and the backward trajectory set points $[p_b(1), \Phi_b(1)]$, $[p_b(2), \Phi_b(2)]$, $\cdots$, $[p_b(r), \Phi_b(r)]$, $\cdots$, this procedure determines a suitable break point on the straight line where deceleration motion starts in order to meet the boundary conditions of the final location.

BR1. [Determine a possible break point]
Determine a set point $p(s)$ on the straight line from the forward and backward search points such that

$$|| \, p(s) - p_b(r) \, || + k_d \cdot || \, \frac{p(s+1) - p(s)}{T} - \frac{p_b(r) - p_b(r-1)}{T} \, ||$$

is minimum over s and r. The number k_d is a weighting factor that measures the slope error between the forward and backward motions. Also, set the loop counter $c = 0$.

BR2. [Apply the MFW algorithm from the set point found in BR1.] Update the loop counter c.
Treating the set point $p(s)$ as an initial position, apply the modified forward search algorithm for the deceleration motion and check whether the algorithm MFW is stopped by the global stopping criterion δ_1.

BR3. [Adjust the location of the break point.]

If the algorithm MFW is stopped by the global stopping constant δ_1,
then decrease s by 1 and go to step BR2,
otherwise go to step BR4.

BR4. [Determine the best break point.]

If the loop counter $c \geq 2$,
then $p(s+1)$ is the best break point, and compute the final joint position, velocity, and acceleration.
otherwise reset the loop counter c to 0, increase s by 1, and go to step BR2.

Step BR1 indicates that the approximated location of the break point is initially selected at a point where each two consecutive set points from the forward and backward search algorithm are very close and the linear velocities between them are very close to each other. The remaining steps are used to correct the approximated location of the break point. Thus we obtain the earliest break point which yields the smallest error at the final search point.

The modified forward binary search algorithm is identical to the algorithm FW except the step FW7 which is modified to accommodate the deceleration motion. The step FW7 is modified in the algorithm MFW as follows:

Algorithm MFW: Given the initial estimate of $\Delta\hat{\lambda}^1(k)$, this algorithm performs a modified iterative forward search of the largest value of $\Delta\lambda(k)$ on the given straight line from the break point to the vicinity of the final position. Steps MFW1 through MFW6 and MFW8 are identical to the steps FW1 through FW6 and FW8 in the Algorithm FW, respectively, except that step MFW3 do not need Eq. (3.67), and ρ_1 and ρ_2 are different from those of the algorithm FW.

MFW7. [Check estimate's constraints and adjust $\Delta\hat{\lambda}^m(k)$]

If the joint inverse kinematics solution satisfies the bounds computed in MFW2,

then set $\Delta\lambda^{high}(k) = \Delta\hat{\lambda}^m(k)$ and $\Delta\hat{\lambda}^{m+1}(k) = \dfrac{\Delta\lambda^{low}(k) + \Delta\lambda^{high}(k)}{2}$, and go to step MFW8,

otherwise set $\Delta\lambda^{low}(k) = \Delta\hat{\lambda}^m(k)$ and $\Delta\hat{\lambda}^{m+1}(k) = \dfrac{\Delta\lambda^{low}(k) + \Delta\lambda^{high}(k)}{2}$, and go to step MFW4.

In algorithm MFW, the ρ_1 and ρ_2 must be chosen such that the former generates the inverse kinematics solution which does not satisfies the constraint limits in MFW2, while the latter generates the inverse kinematics solution which satisfies the constraint limits in MFW2. The forward search algorithm (Algorithm FW) gives us all the control set points on the straight line from the initial position to the break point. This portion of the trajectory constitutes the "acceleration" portion of the given straight line. Treating the chosen break point as an initial position and taking the same final position as before, the remaining control set points of the trajectory are determined using the modified forward binary search algorithm (Algorithm MFW). This portion of the trajectory constitutes the "deceleration" portion of the given straight line.

3.4.2. Existence of Straight Line Trajectory

For the given straight line path, there is no general rule to guarantee the existence of the straight line trajectory with a acceleration portion followed by a

deceleration portion. As discussed in [20], the straight line trajectory between two specified end points may not exist if the average speed of the robot is beyond a certain limit.

In fact, if the constraints on $\Delta\lambda(k)$ from Eqs. (3.18) and (3.24) are not consistent (i.e. $q_{i,min}(k) > q_{i,max}(k)$ or $\tau_{i,min}(k) > \tau_{i,max}(k)$), then the set point on the straight line path, which is based on the smoothness and torque constraints, does not exist at time $t = kT$. In other words, any joint value satisfying the Eqs. (3.18) and (3.24) cannot accommodate the hand position and orientation on the straight line path at time $t = kT$. Then, the algorithm FW or algorithm BW may stop before arriving at the final and initial locations, respectively.

The failure of the algorithm FW or BW can happen for the following cases: (1) when the initial estimate $\Delta\hat{\lambda}^1(k)$ generates the inverse kinematics solution which does not satisfy the smoothness and torque constraints; (2) when the algorithm FW or BW is not completed by the global stopping criterion δ_1. Since either case will generate a large number in the loop counter m, the failure of the algorithm can be detected if the loop counter exceeds a prespecified number of the bisection process. Then a different planning must be used for a given straight line path. Here a procedure (Procedure ESLT) is presented to check the existence of straight line trajectory and to obtain a different trajectory for a given straight line.

Procedure ESLT: Given a straight line path, this procedure addresses the existence of the straight line trajectory and proposes an alternative trajectory planning scheme.

ESLT1. [Check whether the FW algorithm is completed or not.] Identify the failure of the algorithm FW.
(When the loop counter m exceeds a specified number, apply this procedure to a given straight line path.)

If FW search points are completed and stopped by δ_1,
then go to step ESLT2.
otherwise go to step ESLT3.

ESLT2. [Find a break point and apply the MFW algorithm.] Apply the procedure BR and the algorithm MFW, and stop.
(Straight line trajectory planning is completed except for the relaxation portion.)

ESLT3. [Apply the algorithm MFW from the final search point.] Apply the algorithm MFW from the final search point of the algorithm FW to the desired final point.

If the algorithm MFW is completed and stopped by δ_1,
then apply the procedure BR. (from step BR2 to BR4)
(Straight line trajectory planning is completed except for the relaxation portion. But it may not be a near-minimum time trajectory.)
otherwise go to step ESLT4.

ESLT4 [From the final search point of the MFW algorithm, start to apply the FW algorithm.] Take the final search point from the algorithm MFW as the initial point and go to step ESLT1.
(Take the initial estimated $\Delta\hat{\lambda}(k)$ of the final search point as the initial estimated $\Delta\hat{\lambda}(k)$ of the forthcoming algorithm FW in step ESLT1.)

The procedure ESLT generates trajectory set points with a combination of several acceleration and deceleration portions for a given straight line path. As indicated in the procedure ESLT, the resultant trajectory may require a longer traveling time than that of the trajectory with one break point for the given straight line path. The reason is that the procedure is performed several times by the algorithm FW, which is followed by the algorithm MFW from the final search point of the algorithm FW, or vice versa. The application of the procedure ESLT will be shown in Section 6.2.

3.4.3. Relaxation Points

After finding and utilizing the break point in the MFW algorithm, the remaining control set points are obtained from the break point to the final point. We use a subscript g to indicate the information at the final search point. Then, the final search point, $[\, \mathbf{p}(k_g), \Phi(k_g) \,]$ from the algorithm MFW does not reach exactly the desired final position and orientation, $[\, \mathbf{p}(k_{fl}), \Phi(k_{fl}) \,]$ because of the discrete time approximation, the iterative search algorithm and the user-designed stopping constant δ_1. Also, errors exist in the velocity and acceleration between the final search point and the desired final point. If this occurs, then intermediate set points called relaxation points, which satisfy the torque and the smoothness constraints, must be inserted between these two points.

The velocity of the robot in the joint-variable or Cartesian space should be very small at the final search point because of the way in which the break point is computed. If

$$| \ddot{q}_i(k_g) - \ddot{q}_i(k_{fl}) | \leq T \cdot \epsilon_i^J \ ; \ i = 1, \cdots, 6 \tag{3.75}$$

where $\ddot{q}_i(k_g)$ and $\ddot{q}_i(k_{fl})$ are the joint acceleration values at the final search point and the desired final location, respectively, then the time required to meet the acceleration conditions at the final location will be T with position and velocity errors. However, if the above inequalities are not satisfied, then the time required for the robot to meet any of the final conditions must be relaxed to satisfy the smoothness and torque constraints. Two approaches are discussed in the following:

Discrete Time Approach. Since the final search point from the algorithm MFW is within the torque constraint, and the velocity at the final search point is close to zero, we consider the jerk bound only for the relaxation points. It is obvious that

one would like to obtain the smallest value of $\bar{t}$ that is greater than T while generating the set points satisfying the smoothness constraint.

If $\ddot{q}_i(k_{ft}) = \ddot{q}_i(k_g) + w_i$, where w_i is the acceleration difference between the final search point and the desired final point for joint i, then the required time $\bar{t}$ for the relaxation process is:

$$\bar{t} = \max \left\{ \left\lceil \frac{| w_i |}{\epsilon_i^J T} \right\rceil \cdot T \ , \ i = 1, \cdots, n \right\} \stackrel{\Delta}{=} m \ T \qquad (3.76)$$

where $\lceil x \rceil$ is the least integer greater than or equal to x and ϵ_i^J is the jerk bound for joint i. Once the required time $\bar{t}$ is obtained, the relaxation points between $p(k_g)$ and $p(k_{ft})$ can be generated to satisfy the smoothness constraint. The average change of acceleration in time $\bar{t}$ of joint i can be obtained as:

$$\ddot{\bar{q}}_i(\bar{t}) = \frac{w_i}{\bar{t}} = \frac{w_i}{\max \left\{ \left\lceil \frac{| w_i |}{\epsilon_i^J T} \right\rceil \cdot T \ , \ i = 1, \cdots, n \right\}} \qquad (3.77)$$

and the average acceleration for each sampling period T is:

$$\ddot{\bar{q}}_i(\bar{t}) \cdot T = \frac{w_i}{\bar{t}} \cdot T = \frac{w_i}{\max \left\{ \left\lceil \frac{| w_i |}{\epsilon_i^J T} \right\rceil \ , \ i = 1, \cdots, n \right\}} \qquad (3.78)$$

For convenience, we denote the traveling time from the initial location to the break point as FT and from the break point to the final search point as fT. Restarting the time index from the final search point, the acceleration of joint i at $t = jT$ is:

$$\ddot{q}_i(j) = \ddot{q}_i(k_g) + j \cdot \left[\frac{w_i}{\max \left\{ \left\lceil \frac{| w_i |}{\epsilon_i^J T} \right\rceil, \ i = 1, 2, \cdots, n \right\}} \right] \qquad (3.79)$$

where $j = 1, \cdots, m$. During the relaxation period, $\ddot{q}_i(j)$ can be used to deter-

mine $\dot{q}_i(j)$ and $q_i(j)$ and the corresponding values of $p(j)$, $\Phi(j)$, $v(j)$, $\Omega(j)$, $\dot{v}(j)$, and $\dot{\Omega}(j)$. However, we will obtain an erroneous position/orientation, and linear and angular velocity/acceleration at the vicinity of the desired final location because of the relaxation process.

From the above analysis, the total time required to travel the straight line with the specified end points is:

$$\textit{Total traveling time } (k_{fl} \cdot T) = FT + fT + \bar{t} = (F + f + m)\,T \quad (3.80)$$

This total traveling time, which will be denoted as k_f hereafter, is obtained based on all the physical constraints along the straight line with one break point. If more break points are used in planning the straight line path, the average speed of the manipulator in one servo time period will become smaller, which directly requires more servo time intervals to traverse the given straight line. While the discrete time approach generates the error between the actual final point and the desired final point, the joint interpolated trajectory planning scheme, which is now described, can eliminate this error by interpolating the final search point and the desired final point.

Cubic Spline Function Approach. We discuss a joint interpolated trajectory planning scheme by using cubic spline functions. The trajectory planning scheme using cubic spline functions subject to the maximum allowable joint velocity, acceleration and jerk can be found in the literature [34]. Some of the findings in [34] are presented here briefly.

Since the cubic spline function fit is a joint interpolation technique, the continuous time index t instead of the discrete time index k will be used. Let $t_1 < t_2 < t_3 < \cdots < t_{n-2} < t_{n-1} < t_n$ be an ordered time sequence. We set

$t_1 = k_g T$ and $t_n = k_{ft} T$, respectively. For the present development, the argument of q is the actual time t, that is, $q(k_g T)$ is $q(k_g)$ in our notation. Then, from the final search point information and the desired final conditions, we have:

$$q(t_1) = q(k_g T) \; ; \; \dot{q}(t_1) = \dot{q}(k_g T) \; ; \; \ddot{q}(t_1) = \ddot{q}(k_g T) \tag{3.81}$$

and

$$q(t_n) = q(k_{ft} T) \; ; \; \dot{q}(t_n) = \dot{q}(k_{ft} T) \; ; \; \ddot{q}(t_n) = \ddot{q}(k_{ft} T) \tag{3.82}$$

In addition, joint positions $q(t_3)$, $q(t_4)$, $\cdots$,$q(t_{n-2})$ must be specified for the joint trajectory to pass through. Thus, we can assign the joint position values as:

$$q(t_j) = \frac{j-1}{n-1} \left(q(k_{ft}) - q(k_g) \right) + q(k_g) \tag{3.83}$$

where $j = 3, \cdots, n-2$. But $q(t_2)$ and $q(t_{n-1})$ must be free as extra knots for obtaining the cubic polynomial functions.

Let $C_{j,i}(t)$ be a cubic polynomial function defined on the time interval $[t_j, t_{j+1}]$ for joint i. The problem of trajectory interpolation is to spline $C_{j,i}(t)$ for $j = 1, 2, \cdots, n-1$, together such that the required joint position, velocity, and acceleration are satisfied, and the joint position, velocity, and acceleration trajectories are continuous on the entire interval $[t_1, t_n]$. Because $C_{j,i}(t)$ is cubic, the second time derivative $C^2_{j,i}(t)$ can be expressed as

$$C^2_{j,i}(t) = \frac{t_{j+1} - t}{h_j} C^2_{j,i}(t_j) + \frac{t - t_j}{h_j} C^2_{j,i}(t_{j+1}) \tag{3.84}$$

where $i = 1, 2, \cdots, 6$, $j = 1, 2, \cdots, n-1$, and $h_j = t_{j+1} - t_j$. By intergrating Eq. (3.84) twice and utilizing the joint position values, $C_{j,i}(t_j)$ can be derived.

Since the cubic polynomial functions must be constrained by the maximum velocity, acceleration, and jerk limits for each joint, we have:

$$
\begin{aligned}
&| \, C^1_{j,i}(t) \, | \leq \epsilon_i^v \\
&| \, C^2_{j,i}(t) \, | \leq \epsilon_i^a \\
&| \, C^3_{j,i}(t) \, | \leq \epsilon_i^J.
\end{aligned} \tag{3.85}
$$

where $C^3_{j,i}(t)$ denotes the third time derivative of the cubic polynomial function. We denote the total traveling time as h_t, where $h_t = h_1 + \cdots + h_n$. By developing an optimizing algorithm, the total traveling time h_t for the relaxation portion has been optimized under the constraints of maximum velocity, acceleration, and jerk limits.

Due to the cubic spline function fit, there are no errors in joint position, velocity and acceleration between the actual final point and the desired final point. However, we must calculate the coefficients of the cubic spline functions for the relaxation portion of the straight line trajectory. The resultant trajectory set points, obtained by evaluating the cubic spline functions, yield joint values, which satisfy the smoothness constraints but are not on the desired straight line path. Also, the total number of servo time intervals for the relaxation portion can be determined from dividing the total time h_t by the servo time interval T. Usually, more servo time intervals than m will be required at the expense of fitting the boundary conditions at the final location. A more detailed derivation can be found with related simulation results in the reference [34].

After getting $\mathbf{q}(k)$, $\mathbf{p}(k)$, and $\Phi(k)$ from the search algorithm followed by the relaxation process, they can be used to determine $\mathbf{v}(k)$ and $\Omega(k)$ using the kinematic equation in Eq. (2.5). In summary, the whole procedure to obtain the straight line trajectory can be stated as follows:

Overall Trajectory Planning Procedure

Step 1: Given a straight line path specified by two end points, apply the algorithms FW and BW.
If the algorithms FW and BW are completed successfully,
then go to step 2,
otherwise, go to step 3.

Step 2: Apply the procedure BR to obtain the servo control points of the acceleration portion and the deceleration portion, and go to step 4.

Step 3: Apply the procedure ESLT to the given straight line and go to step 4.

Step 4: Obtain the relaxation points from the discrete time approach or the cubic spline function approach.
(Trajectory planning has been completed.)

A block digram showing the generation of Cartesian control set points of a given straight line is shown in Figure 3.2.

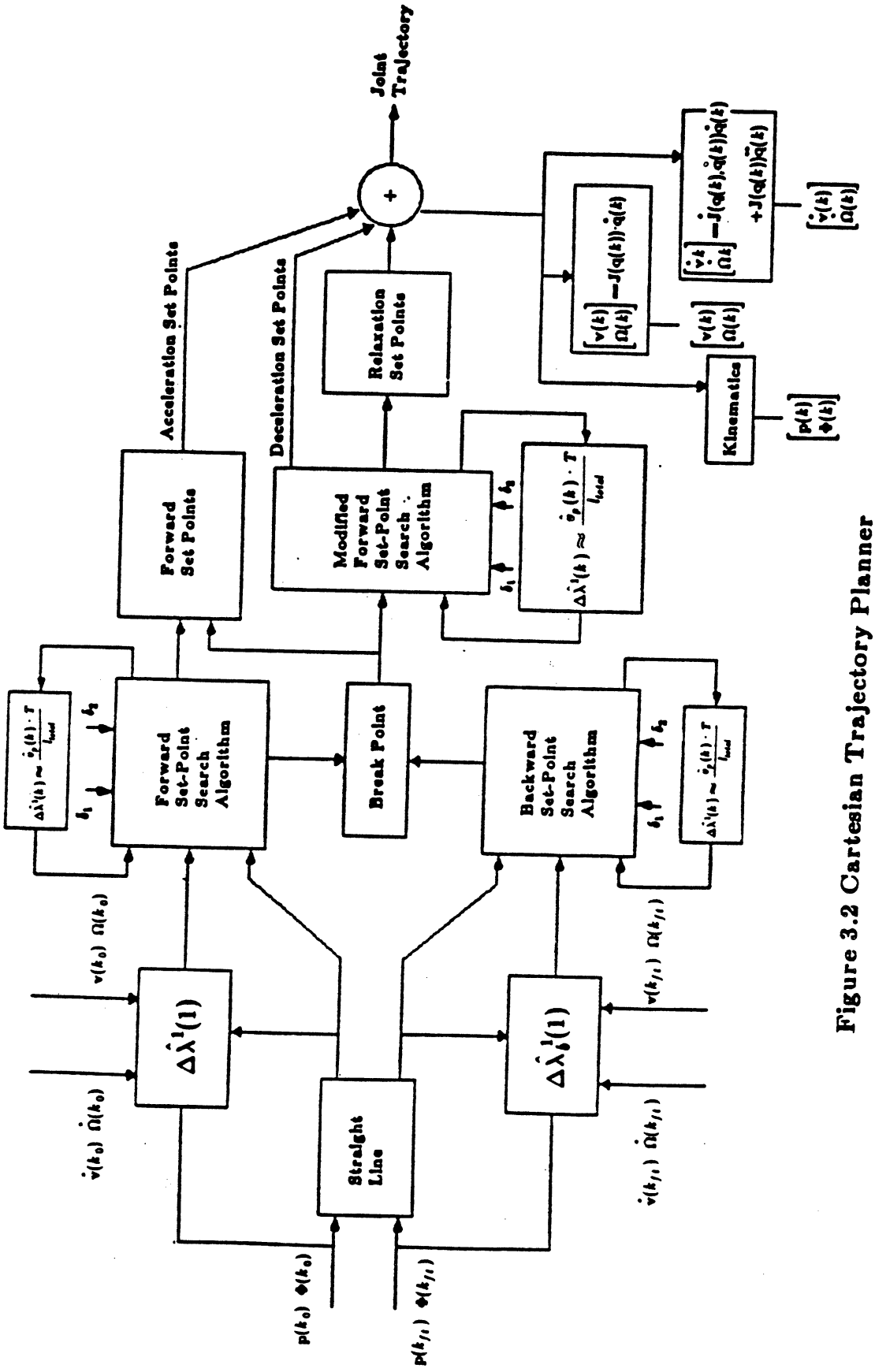


Figure 3.2 Cartesian Trajectory Planner

3.5. Summary

This chapter presents a scheme in straight line trajectory planning subject to the various constraints. The scheme is based on the discrete time trajectory analysis. A real time servo interval instead of a normalized time interval is used for planning the manipulator trajectory. Because of split constraints in joint and Cartesian coordinate systems, the set points are determined by maximizing the path length ratio between two consecutive Cartesian control set points subject to the smoothness and torque constraints and the straight line requirement.

An iterative forward search algorithm is developed to determine the control set points from the initial position to the break point on the straight line. A modified forward search algorithm is then developed to determine the control set points from the break point to the vicinity of the final point. The final search point is found to be very close to the desired final point. However, relaxation points, which do not satisfy the straight line requirement, may be necessary if these two points exhibit a large acceleration difference. Thus, the straight line trajectory is composed of three segments: the acceleration portion, the deceleration portion, and the relaxation portion. The existence of the straight line trajectory is also discussed.

Computationally, the proposed trajectory planning scheme provides a simple method of determining the control set points, which is different from conventional methods that require the determination of the joint interpolation function followed by the evaluation process. Simulation results for planning a straight line trajectory are presented in Chapter 6. Finally, Chapter 5 describes how the resultant trajectory set points are used for the Cartesian space control.

CHAPTER 4

PLANNING OF COLLISION-FREE MANIPULATOR PATH

4.1. Introduction

In this chapter, we discuss a method for obtaining collision-free paths of two robots in a common workspace. The planning of collision-free paths of two robots is considered in two classes: one robot is stationary and the other robot is moving; and two robots are moving simultaneously. The original pre-planned paths of two robots are composed of straight line paths. It is assumed that no collisions occur at the initial and final locations of the straight line paths between the robots. It is also assumed that the straight line trajectories of two robots are planned by the trajectory planner discussed in Chapter 3.

Notations for geometric analysis are presented and several performance measures are discussed in Section 4.2. Using a sphere model to represent the wrists of each robot, a collision-free path for a moving robot in the presence of a stationary robot is found which consists of connected straight line segments. For the class of two moving robots, various collision situations are identified and discussed in Section 4.3, where notions of collision map and time scheduling are introduced. Two procedures are developed to obtain collision-free paths and/or trajectories in Section 4.4. Finally, summaries are presented in Section 4.5.

4.2. Collision-Free Path for the Case of a Stationary Robot

4.2.1. Overviews and Notations

In this section, the collision situation between a stationary robot and a moving robot is investigated using a geometric analysis technique. The analysis assumes that a straight line trajectory is known for the moving robot and that the wrist of each robot is modeled by a single sphere. Several performance measures are considered and necessary notations are presented.

In Figure 4.1, a moving robot (say robot 2) having a wrist of radius r_2 collides with the wrist of radius r_1 of a stationary robot (say robot 1) during the time interval from $t = mT$ to $t = nT$. That is, no collision occurs before $t = mT$ and after $t = nT$. The original straight line path WZ of robot 2 must be replanned to avoid wrist collision with robot 1. The curve from point X to point Y is a curve, which robot 2 must follow for the collision avoidance with minimum deviation from the original path WZ. We call this a collision-free curve. Let us denote Q as the origin of the sphere of robot 1 with respect to a fixed global coordinate frame. Then, the distance from Q to the origin of the sphere of robot 2 must be at least $r_1 + r_2$ for the wrist collision avoidance. The locus of the distance of $r_1 + r_2$ can be realized as a curve (or an arc) of the "pseudo-sphere" (or colliding sphere) surface of radius $r_1 + r_2$.

Depending on the performance measure selected, this curve will be approximated by finding and connecting via points, which can be determined subject to the allowable deviation error from the collision-free curve. Straight line segments will be used to approximate the curve. Several performance measures can be considered for measuring the deviation of proposed paths from the path WX, XY (along the

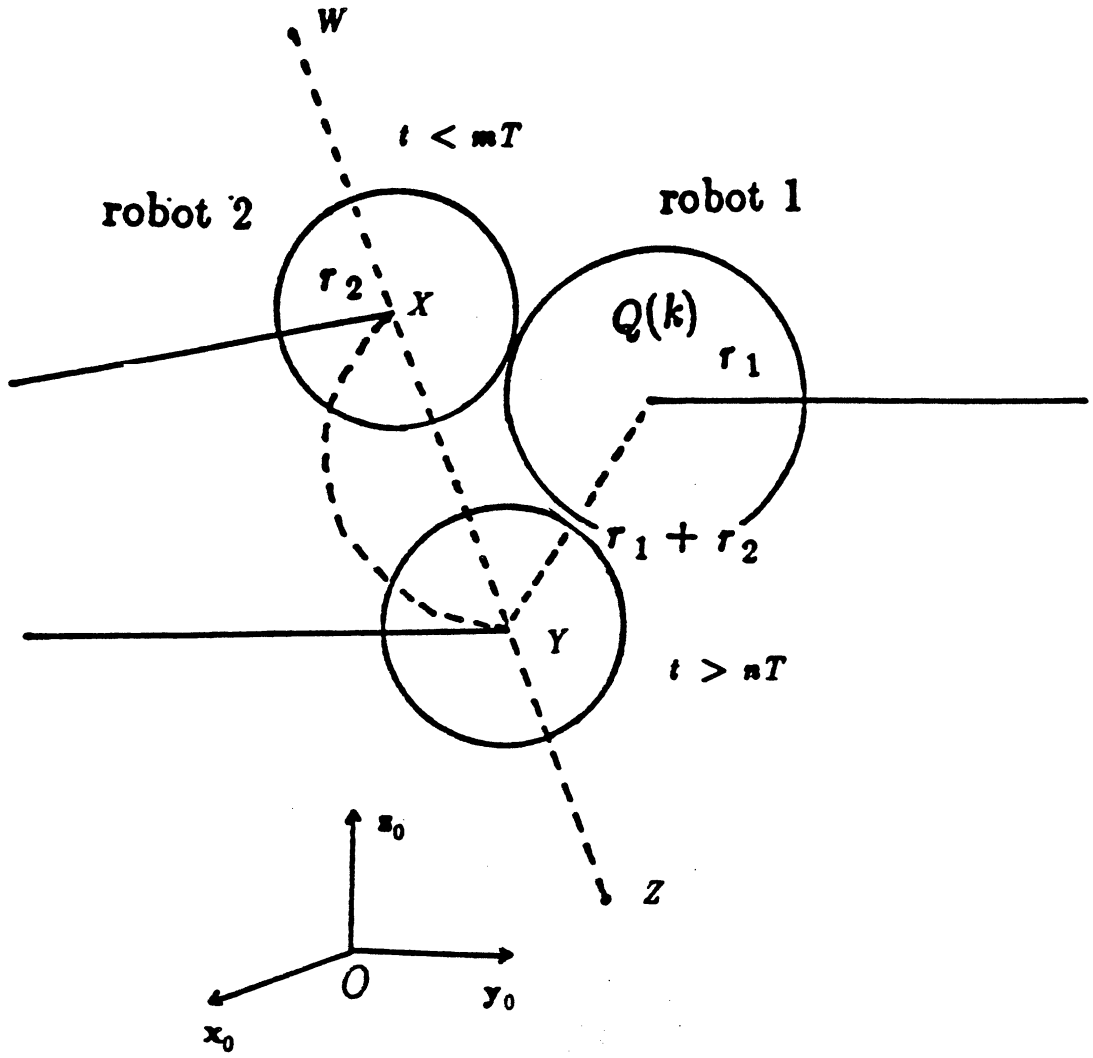


Figure 4.1 Stationary Robot Avoidance

collision-free curve), and YZ.

They are:

(1) Minimum Deviation from the Collision-Free Curve : Minimization of the maximum deviation error from the collision-free curve XY while constraining the path to lie on the portion of original path between WX and YZ

(2) Minimum Deviation from the Original Path: Minimization of the maximum deviation error from the path composed of the straight line segments WX, YZ and the curve segment XY

(3) Minimum Spanned Area: Minimization of the area enclosed by a collision-free path and the original path

(4) Least Squared Distance Error: Least squared distance error calculated at each servo interval between the original path and a collision-free path

(5) Minimum Energy: Minimization of energy for traveling along a collision-free path

(6) Minimum Time: Minimization of traveling time for a collision-free path

Among the various possible measures, the first two measures (we call them Measure (1) and Measure (2)), are utilized to obtain a collision-free path in later sections. The application of some other measures can be found in the literature [8,19,34,38]. In order to carry out the geometric analysis for collision-free path planning, necessary notations of a plane, a line, and the direction number of a straight line are presented in the following.

A plane in a 3 dimensional space is described as:

$$a_1 x + b_1 y + c_1 z = d_1 \tag{4.1}$$

In general, a line is generated by two intersecting planes in the 3 dimensional space. Since a plane is represented by Eq. (4.1), a line can be represented as:

$$\begin{aligned} a_1 x + b_1 y + c_1 z &= d_1 \\ a_2 x + b_2 y + c_2 z &= d_2 \end{aligned} \tag{4.2}$$

The direction number of the line described in Eq. (4.2) can be written as (see Appendix E):

$$[\, b_1 c_2 - c_1 b_2,\ c_1 a_2 - a_1 c_2,\ a_1 b_2 - b_1 a_2 \,] \tag{4.3}$$

The direction number of the line will be used in the next section to analyze the geometry of the collision situation.

A point Q_1 is represented with its coordinates as $Q_1(x_{Q1}, y_{Q1}, z_{Q1})$,[4] throughout this chapter, where the subscript $Q1$ denotes the point Q_1. The distance d between the points Q_1 and Q_2 is obtained from the norm of the vector subtraction and represented as:

$$d = \| \vec{OQ}_1 - \vec{OQ}_2 \| \tag{4.4}$$

where the point O is the origin of the global coordinate frame.

4.2.2. Planning of Collision-Free Path by Measure (1)

In this section, the notations in Section 4.2.1 are utilized to plan a collision-free path for a moving robot in the presence of a stationary robot. From the measure (1), via points around the curve are selected so that a path that passes through these via points will approximate closely the collision-free curve XY in Figure 4.1. Tangential cut planes around the pseudo sphere are derived to determine these via

[4]We use the notation $Q_1(x_{Q1}, y_{Q1}, z_{Q1})$ to represent a point Q_1 in the Cartesian space expressed with respect to the global coordinate system with the coordinate (x_{Q1}, y_{Q1}, z_{Q1}).

points. As a result of connecting these via points, the collision-free curve XY is approximated by connected straight line segments.

In Figure 4.2, a collision situation is shown, where the point M is a point on the original path of the moving robot (robot 2) at time $t = mT$, the point N is a point on the original path of the moving robot at time $t = nT$, the point Q is the origin of the wrist model of the stationary robot and the point O is the origin of the global coordinate frame. We assume that the time $t = mT$ is the time at which a potential collision would first occur and the time $t = nT$ is the time at which a potential collision would last occur. Since the potential collision would occur at time between $t = (m-1)T$ and $t = mT$, it is desirable to find a potential collision starting location. Similarly, a potential collision ending location must be found. Let us denote the path of robot 2 as:

$$\begin{aligned} h_1 x + h_2 y + h_3 z &= h_4 \\ k_1 x + k_2 y + k_3 z &= k_4 \end{aligned} \tag{4.5}$$

and the equation of the pseudo-sphere with radius $r_1 + r_2$ discussed in Section 4.2.1 as:

$$(x - x_Q)^2 + (y - y_Q)^2 + (z - z_Q)^2 = (r_1 + r_2)^2 \tag{4.6}$$

where $Q(x_Q, y_Q, z_Q)$ is the origin of the stationary robot 1. Then, the actual collision starting point A and the actual collision ending point B can be obtained by solving Eqs. (4.5) and (4.6) simultaneously. The point A can be interpreted as the point at which robot 2 must deviate from its original path for collision avoidance and the point B as the point after which robot 2 must continue to move on the original path. We obtain the tangential plane to the pseudo-sphere at the point $A(x_A, y_A, z_A)$, called TP_A, which is represented as:

$$(x-x_A)(x_A-x_Q)+(y-y_A)(y_A-y_Q)+(z-z_A)(z_A-z_Q)=0 \tag{4.7}$$

Similarly, the tangential plane to the pseudo-sphere at the point $B(x_B, y_B, z_B)$, called TP_B, is obtained as:

$$(x-x_B)(x_B-x_Q)+(y-y_B)(y_B-y_Q)+(z-z_B)(z_B-z_Q)=0 \tag{4.8}$$

The intersection line between these two tangential planes is obtained by Eqs. (4.7) and (4.8). That is, the intersection line is described by:

$$\begin{aligned}(x-x_A)(x_A-x_Q)+(y-y_A)(y_A-y_Q)+(z-z_A)(z_A-z_Q)&=0\\(x-x_B)(x_B-x_Q)+(y-y_B)(y_B-y_Q)+(z-z_B)(z_B-z_Q)&=0\end{aligned} \tag{4.9}$$

Let us use C_1 to denote the point from A to the intersection line and C_2 to denote the point from B to the intersection line, which are all in the perpendicular direction to the intersection line. That is, if we use the notation SS' to indicate the intersection line, then $\vec{AC}_1$ is perpendicular to the line SS' and $\vec{BC}_2$ is also perpendicular to the line SS'. It can be shown that the points C_1 and C_2 are the same points if two planes are the tangential planes of a sphere.

By definition, the vector $\vec{QA}$ is perpendicular to the tangential plane TP_A and the vector $\vec{QB}$ is perpendicular to the tangential plane TP_B. Since the intersection line SS' is on the planes TP_A and TP_B, it is also perpendicular to the vectors $\vec{QA}$ and QB. This results from the fact that the line SS' is in the same direction as the normal vector to the plane which is uniquely determined from the point Q, and vectors $\vec{QA}$ and $\vec{QB}$. We denote this plane as TP_Q. Then any lines from the points A and B, which are perpendicular to the line SS' and meet the line SS', exist on the plane TP_Q. Therefore, the points C_1 and C_2 will coincide at a point C, which is the intersection point between the line SS' and the plane TP_Q. From

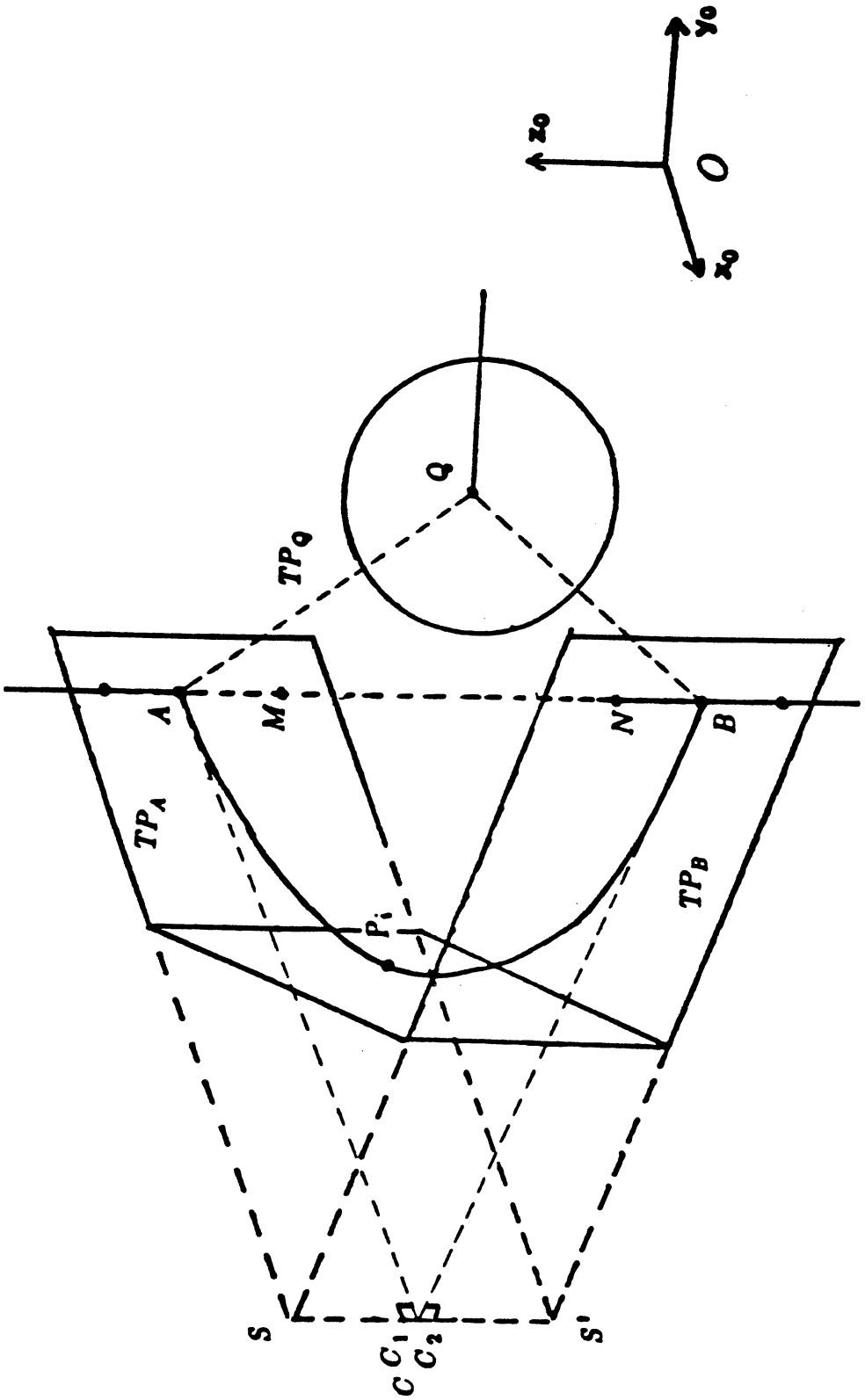


Figure 4.2 Tangential Cut Planes

now on, we will utilize the tangential lines instead of the tangential planes due to the fact that all the via points are on the plane TP_Q.

Since the direction number $[\alpha_S, \beta_S, \gamma_S]$ of the line SS' is obtained from Eqs. (4.3) and (4.9) as:

$$\begin{aligned} \alpha_S &= (y_Q - y_A)(z_Q - z_B) - (z_Q - z_A)(y_Q - y_B) \\ \beta_S &= (z_Q - z_A)(x_Q - x_B) - (x_Q - x_A)(z_Q - z_B) \\ \gamma_S &= (x_Q - x_A)(y_Q - y_B) - (y_Q - y_A)(x_Q - x_B) \end{aligned} \tag{4.10}$$

the plane TP_Q is described as:

$$\alpha_S(x - x_Q) + \beta_S(y - y_Q) + \gamma_S(z - z_Q) = 0 \tag{4.11}$$

The point C can be determined by solving the line SS' equation (Eq. (4.9)) and the plane TP_Q equation (Eq. (4.11)) together. However, since the point C is on the plane TP_Q, a simpler method will be utilized to find the point.

In fact, the collision-free curve, which is the intersected curve of the pseudo-sphere and the plane TP_Q, can be obtained by solving Eqs. (4.9) and (4.11) simultaneously. If the path of robot 2 described by Eq. (4.5) becomes closer to the origin of the pseudo-sphere, then the deviation of the via point C from the collision-free curve will increase. Then the approximation of the path (A to B via C) to the collision-free curve (on the pseudo-sphere) will be difficult to justify. In the ultimate case, if the path of robot 2 passes through the origin of the sphere of robot 1, then the two tangential planes TP_A and TP_B will not intersect each other. Thus, additional via points will be required to avoid these situations and to get a better approximation. Two via points can be obtained by inserting an intermediate tangential plane at a point P_i as in Figure 4.2.

The determination of via points requires finding tangent points on the collision-free curve, where tangential planes or tangential lines to the pseudo-sphere can be found. Since one intermediate tangential line will generate two via points, two intermediate tangential lines will generate three via points, and so on, the selection of c intermediate via points (the number of the via points is determined from the allowable deviation error of the path) to approximate the collision-free curve will require $c - 1$ intermediate tangential lines. The points on the collision-free curve, where the tangential lines to the curve will be found, are denoted as P_i, where $i = 1, 2, ..., c - 1$ as in Figure 4.3. The points P_i can be obtained by rotating the point A around the normal vector of the plane TP_Q by an angle of $\frac{i\phi}{c}$, where the angle ϕ is the angle between $\vec{QA}$ and $\vec{QB}$ and $i = 1, 2, ..., c - 1$. Notice that the points P_i when $i = 0$ and $i = c$ correspond to the points A and B respectively.

Let us denote the normal unit vector of the plane TP_Q as $\vec{ON}_Q$. The vector $\vec{ON}_Q = (n_x, n_y, n_z)^T$ is obtained as:

$$\vec{ON}_Q = \frac{\vec{QA} \times \vec{QB}}{||QA \times QB||} \tag{4.12}$$

where $\times$ denotes the vector cross product. Then, the tangent points can be found as [7,42]:

$$\begin{aligned} \vec{OP}_i &= e^{R_N \frac{i\phi}{c}} \vec{QA} + \vec{OQ} \\ &= \left\{ I\cos(\frac{i\phi}{c}) + \vec{ON}_Q \vec{ON}_Q^T(1 - \cos(\frac{i\phi}{c})) + R_N \sin(\frac{i\phi}{c}) \right\} \cdot \vec{QA} + \vec{OQ} \end{aligned} \tag{4.13}$$

where I is a 3×3 identity matrix, $i = 1, 2, ..., c - 1$, O is the origin of the global coordinate frame and

$$R_N = \begin{bmatrix} 0 & -n_z & n_y \\ n_z & 0 & -n_x \\ -n_y & n_x & 0 \end{bmatrix} \tag{4.14}$$

To find the via points, we first find the tangential lines at P_i to the collision-free curve, and then determine the intersection points. Since every via point is on the same plane TP_Q, they can be obtained by adding two consecutive $\vec{QP}_i$ vectors and scaling appropriately. A via point V_i is obtained by adding two vectors $\vec{QP}_i$ and $\vec{QP}_{i-1}$ and taking the magnitude as:

$$|| \vec{QV}_i || = (r_1 + r_2) \cdot \sec(\frac{\phi}{2c}) \tag{4.15}$$

where $i = 1, 2, ..., c - 1$. That is, the vector $\vec{QV}_i$ is represented as:

$$\vec{QV}_i = (\frac{\vec{QP}_i + \vec{QP}_{i-1}}{|| \vec{QP}_i + \vec{QP}_{i-1} ||}) \cdot (r_1 + r_2) \cdot \sec(\frac{\phi}{2c}) \tag{4.16}$$

where $i = 1, 2, ..., c - 1$. Therefore, the vector $\vec{OV}_i$ is:

$$\vec{OV}_i = \vec{QV}_i + \vec{OQ} \tag{4.17}$$

and the deviation error of the via point V_i from the collision-free curve is:

$$d = (r_1 + r_2) \cdot (sec(\frac{\phi}{2c}) - 1) \tag{4.18}$$

If the maximum allowable deviation error is d_1, then we have:

$$(r_1 + r_2) \cdot (sec(\frac{\phi}{2c}) - 1) \leq d_1 \tag{4.19}$$

Thus, the required number of via points c for the maximum allowable error d_1 is obtained as:

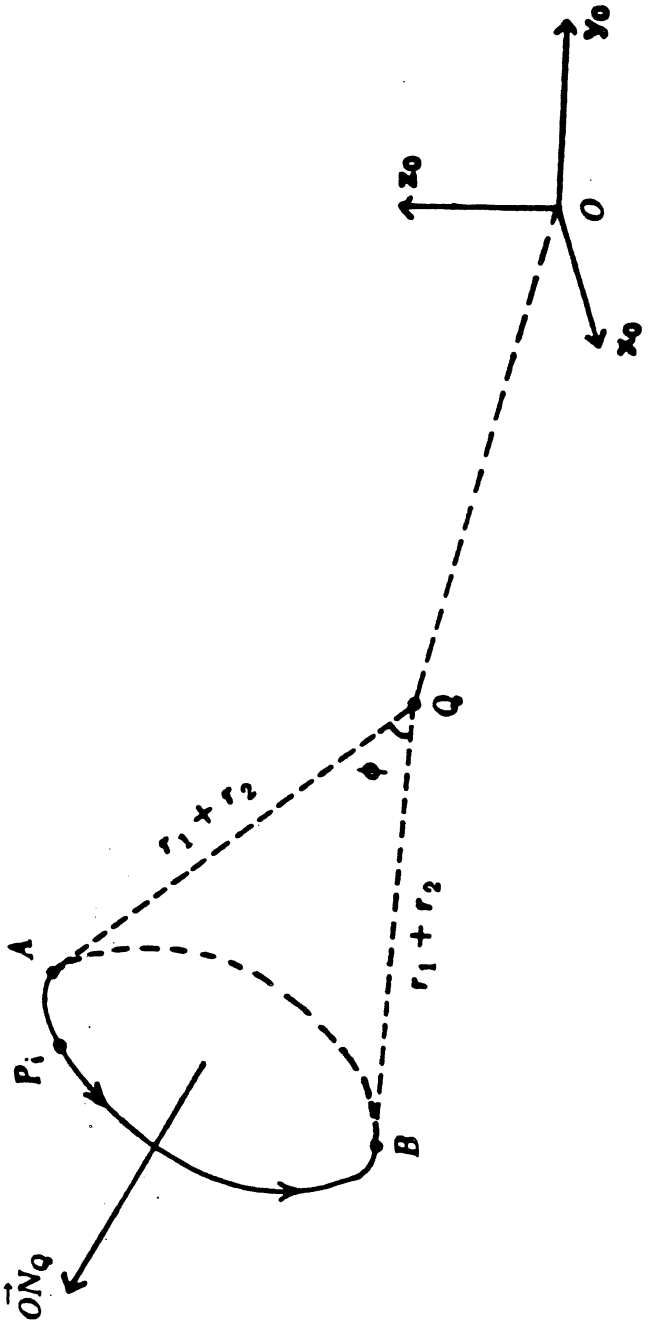


Figure 4.3 Rotation of a Vector

$$c \geq \frac{\phi}{2} \cdot \frac{1}{\cos^{-1}\left(\frac{r_1 + r_2}{r_1 + r_2 + d_1}\right)} \tag{4.20}$$

From Eq. (4.20), it is found that if the maximum allowable deviation error d_1 approaches to 0, then the required number of via points, c, will approach infinity, which results in the collision-free path the same as the collision-free curve on the pseudo sphere. The decision of how many via points must be used to approximate the pseudo-sphere curve is up to the allowable deviation error and the path accuracy for the collision avoidance purpose, and the user has the freedom to decide it.

4.2.3. Planning of Collision-Free Path by Measure (2)

From the Measure (2) in Section 4.2.1, a collision-free path is derived and obtained around the original path and the collision-free curve by approximating the path and the curve within the allowable deviation error. Given the allowable deviation error from the curve, the number of necessary via points are calculated to derive the collision-free path.

In Figure 4.4, assume that d_2 is the allowable deviation error of the collision-free path from the collision-free curve and c is the required number of via points corresponding to the deviation error d_2. Also, point C is the intersection point between the tangential plane to the pseudo-sphere at point P_1 and the original robot 2 path. Similarly, point D is the intersection point between the tangential plane to the pseudo-sphere at point P_{c+1} and the original robot 2 path. The angle between $\vec{QA}$ and $\vec{QB}$ is denoted by ϕ as before. Since the angle between $\vec{QA}$ and $\vec{QP}_1$ is denoted by δ and the radius of the pseudo-sphere is $r_1 + r_2$, we have:

$$d_2 = (r_1 + r_2) \cdot (1 - \cos\delta) \tag{4.21}$$

which leads to

$$\delta = \cos^{-1}\left(\frac{r_1 + r_2 - d_2}{r_1 + r_2}\right). \tag{4.22}$$

Once the maximum allowable deviation error, d_2, is given, the angle δ can be obtained and used to find the points P_1 and P_{c+1} by using Eq. (4.13) with replacement of the angle of the trigonometric function by δ and $\phi - \delta$, respectively. Thus tangential planes at points P_1 and P_{c+1} can be derived and used to find the points C and D. The angle between $\vec{QP}_1$ and $\vec{QP}_{c+1}$ is denoted by ϕ' and can be obtained as:

$$\phi' = \phi - 2 \cdot \delta \tag{4.23}$$

Recall that the deviation error from c intermediate via points is obtained from Eq. (4.18). Since the maximum deviation error must be equal to or less than d_2, we have:

$$(r_1 + r_2) \cdot \sec\left(\left(\frac{\phi'}{2c}\right) - 1\right) \leq d_2 \tag{4.24}$$

Combining Eqs. (4.22)-(4.24), we have:

$$c \geq \frac{\phi'}{2} \cdot \frac{1}{\cos^{-1}\left(\frac{r_1 + r_2}{r_1 + r_2 + d_2}\right)} \tag{4.25}$$

From Eq. (4.25), it is found that if d_2 approaches to 0, then, the number of required via points c will approach infinity, which results in the collision-free path the same as the collision-free curve on the pseudo-sphere.

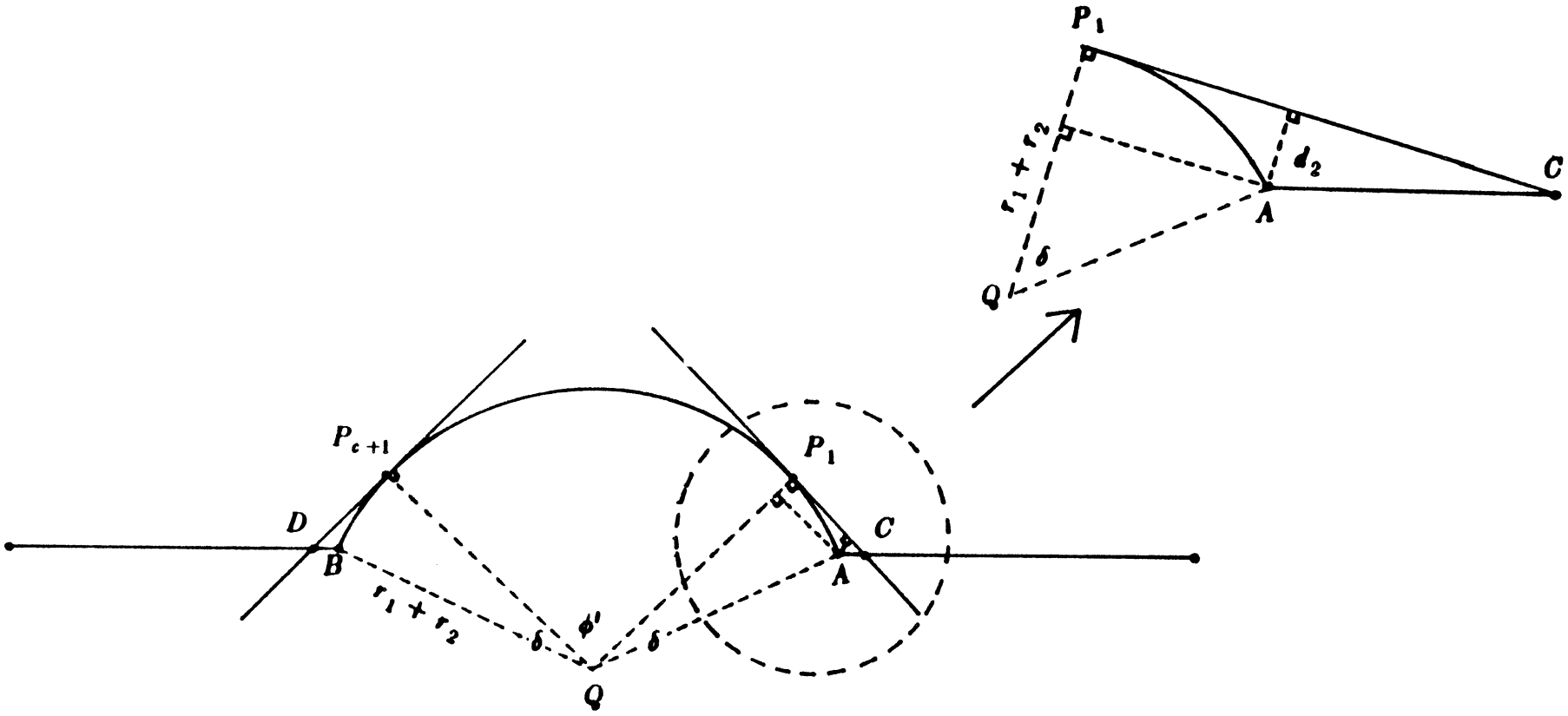


Figure 4.4 Geometric Analysis For Deviation Error

4.2.4. Comparisons and Discussions

Methods for obtaining collision-free paths for a stationary robot and a moving robot have been discussed by approximating the collision-free curve with connected straight line segments. Depending on the selected measure, the number of required via points are derived in Eqs. (4.20) and (4.25) for the given allowable deviation errors. For a specified maximum deviation error, it is found that the Measure (2) requires fewer via points for the approximated collision-free path. Ultimately, if the path of robot 2 passes through the origin of the pseudo-sphere, then the difference of the required number of via points will be the maximum for the specified allowable deviation error.

Another aspect is the total required traveling time of the collision-free path. Since the Measure (2) requires fewer via points for the specified allowable deviation error than the Measure (1) does, the total traveling time for the path from the Measure (2) is likely to be shorter than that from the Measure (1). Since the robot must stop and go at each of the intersection points of c connected straight line segments, the traveling time in both Measures will be longer when a more accurate approximation to the collision-free curve is obtained.

4.3. Time Scheduling and Collision Map

In this section, we discuss notions of time scheduling and collision map for finding the collision-free trajectory for two moving robots. The time scheduling is a procedure to modify and reschedule the traveling speed along the pre-planned trajectory for the purpose of collision avoidance. In a broader sense, time scheduling also implies time delaying the motion of one of the robots. Thus, the robot motion may slow down or be delayed as a result of the time scheduling of the trajectory.

The collision map is a two dimensional figure, in which we can incorporate both the path and trajectory information of two moving robots simultaneously. The time scheduling procedure is used with the collision map in obtaining collision-free trajectories for two moving robots.

4.3.1. Case 1 and Case 2 Collision Situations

As described in Section 2.5, methods in obtaining the collision-free path may vary depending on the various collision situations when two robots are moving on the straight line paths simultaneously. Since trajectory information from Chapter 3 is utilized, the discrete time index is used here instead of the continuous time index. For example, we use k_f and k_0 instead of t_f and t_0 to denote the final arrival time and the initial starting time of the robot, respectively. Only cases 1 and 2 as discussed in Section 2.5 are considered in this section.

In case 1, the final arrival time k_f of robot 2 can be relaxed but the original straight line path cannot be altered for the collision avoidance purpose. Thus, we can only change the speed or delay the motion of robot 2 along the original path to avoid the collision. Since the change of the robot speed can only be accomplished by modifying the trajectory information, the time scheduling procedure will be applied to obtain the collision-free trajectory of robot 2. The trajectory set points from the trajectory planner are obtained based on the physical constraints of the robot, thus the time scheduled trajectory is obtained with the smoothness and torque constraints taken into consideration. Details about the time scheduling for case 1 collision situation are discussed in Section 4.4.1.

In case 2, we discuss a collision-free path of robot 2 in the following categories: (1) when time scheduling is applied without any path modification; (2) when only

path modification is applied without considering the trajectory information of two robots; (3) when path modification and time scheduling are applied simultaneously. In category (1), a collision-free path can be found exactly in the same way as in case 1. If a solution from category (1) is not adequate because a fairly large time delay is required and speed reduction is not appropriate for the collision avoidance (we will show this later), then we can consider a solution in categories (2) and (3). In category (2), a collision-free path can be found through a geometric analysis. To reduce the unnecessary path deviation from category (2), path modification and time scheduling can be applied simultaneously. In category (3), we apply path modification and time scheduling at the same time. There exists an infinite number of collision-free paths and trajectories in this category. A method is presented for planning a collision-free path with path modification and time scheduling of the trajectory subject to the performance measures selected. Detailed derivation will be presented in Section 4.4.2.

In the next section, the general concepts of time scheduling of a straight line trajectory are described. These concepts are then used to obtain the collision-free trajectory for cases 1 and 2.

4.3.2. Time Scheduling of Straight Line Trajectory

In Figure 4.5, assuming that a straight line path whose trajectory set points are obtained from the trajectory planner discussed in Chapter 3, A is the initial location, C is the final location, k_0 is the initial time and k_f is the final time of the robot. We further assume that the given path can be planned with a break point. By applying the algorithms FW and MFW together with the procedure BR to the given straight line path, we obtain the trajectory set points along the path with an

acceleration portion and a deceleration portion. We denote this break point B at time k_B on the path. Then, during the time interval $[k_0, k_B]$, the manipulator hand is in the accelerating motion and during the time interval $[k_B, k_f]$, the manipulator hand is in the decelerating motion. Since the set points in the relaxation portion of the trajectory are not on the given straight line path, we shall plan the trajectory without the relaxation portion of the trajectory.

From the procedure BR, the break point B can be selected uniquely such that the position and Euler angle errors between the final search point and the desired final point are the smallest. In order to reduce the speed of the robot or increase the traveling time along the path, additional break points are used at appropriate locations on the path. Thus, referring to Figure 4.5, if the robot starts the decelerating motion from a point D, which is located in the acceleration portion of the straight line $\overline{AC}$, then the decelerating motion will end up at some point E after applying the algorithm MFW with modified stopping criterion and a different starting point D, which results in longer traveling time of the robot from point A to point E than that from the original trajectory information. As a result, the robot will be in the accelerating motion on the portion $\overline{AD}$ and in the decelerating motion on the portion $\overline{DE}$.

From point E in Figure 4.5, we can determine another break point F for an acceleration portion $\overline{EF}$ and a deceleration portion $\overline{FC}$. That is, if we use 3 break points (points D, E, and F) for the trajectory planning of the straight line path, then the location of the first break point D will determine the whole time scheduling with the locations of the break points E and F.

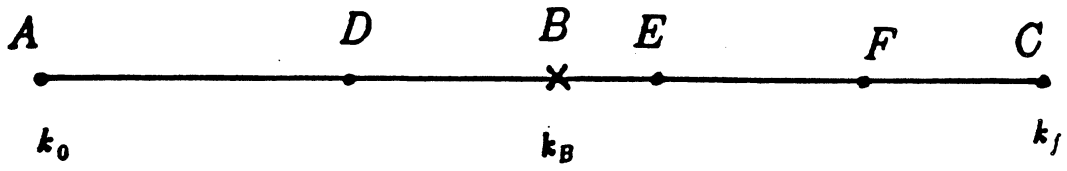


Figure 4.5 Initial Path and Trajectory Planning

If the point E is taken as a break point, then the segment $\overline{AE}$ in Figure 4.5 can be regarded as a single segment and the trajectory planning of the segment corresponds to the time scheduling of the original trajectory. Thus, an immediate objective of time scheduling is to find segments such that the whole trajectories of the segments can avoid the potential collisions. Also, the final portion of the straight line must be in the decelerating motion to reduce the final position and Euler angle errors. Thus, the total number of break points on a straight line path must be *odd*.

The trajectory planning of a straight line path with one break point results in near-minimum time for traversing the path as discussed in Chapter 3. Thus, if we have more segments for a path, then the total traveling time will be greater than that with a break point because the average traveling speed of the manipulator hand will be much slower than before. However, we must also note that the number of all the possible combinations of the acceleration portion and the deceleration portion may be very large. In the limiting case, we can apply the algorithm FW and the algorithm MFW to every two trajectory set points. Hence, the time scheduling of a straight line trajectory is a procedure in achieving the required "time delay" or "speed reduction" of the robot motion for the collision avoidance purpose.

In the next section, we shall present a collision map which incorporates the location and the corresponding servo time information of the robot into a two dimensional figure. This will be used in obtaining the collision-free path or collision-free time scheduling of the robot in Section 4.4.

4.3.3. Collision Map

Since the collision-free paths of two moving robots are related not only to the path, but to the trajectory information, a notion of a collision map is introduced in this section. It is useful in obtaining the collision-free paths or the collision-free trajectories of the two moving robots.

The collision map is obtained from the path and trajectory information of two moving robots. From the trajectory planner in Chapter 3, the traveled length $\lambda(k) \cdot l_{total}$ and the corresponding servo time instant $t = kT$ of robot 2 can be obtained along a straight line path. Using this information, a curve, which relates the traveled length with corresponding servo time instant of robot 2, can be drawn. See Figure 4.6, where the vertical axis is the traveled length and the horizontal axis is the servo time.

Two moving robots have the potential for colliding with each other under the original trajectory information, if there is a range of collision lengths where the path of robot 2 is within the colliding range of a point on the path of robot 1. The union of these collision lengths at the collection of servo time instants determining the points on the path of robot 1 can be drawn as a connected region. See Figure 4.6. More precisely, the collision length at time k corresponds to all points on the path for robot 2 that are within $r_1 + r_2$ of the point that lies on the path of robot 1 at time k. If the traveled length versus servo time curve touches or crosses the region, it indicates that a potential collision of the wrists of robot 1 and robot 2 exists.

Now, we present a method to obtain the collision region in more detail. Assume that two robots are moving in a straight line fashion as shown in Figure 4.7, where robot 1 has to move from point A_1 to point B_1 and robot 2 has to move

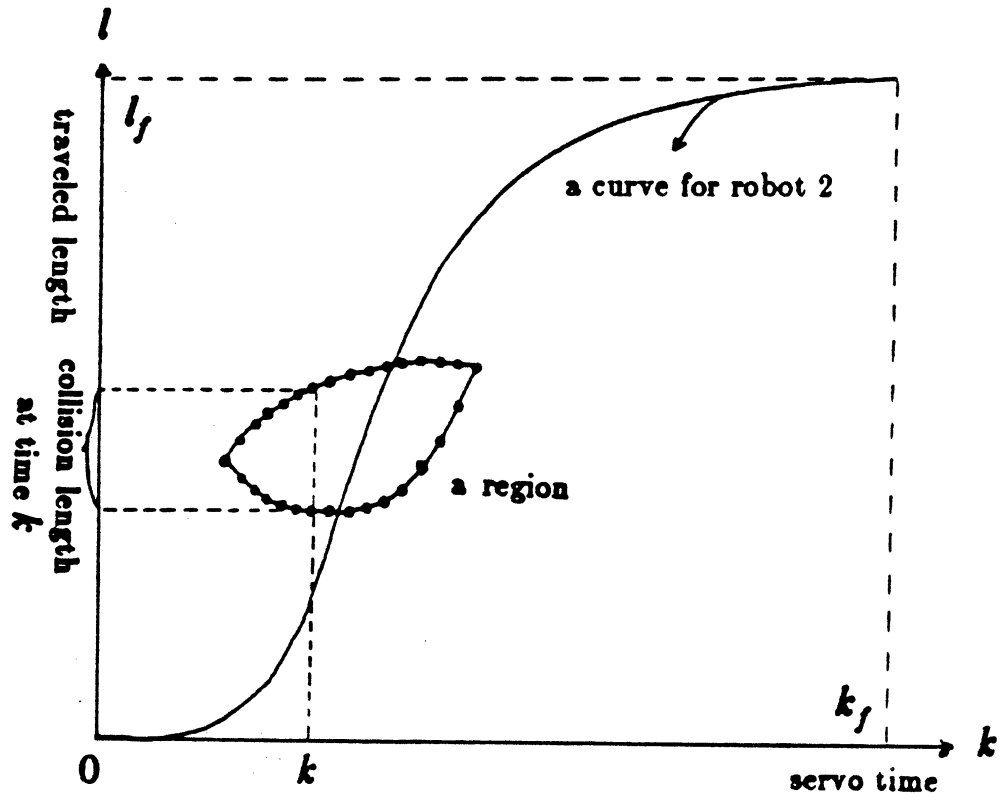


Figure 4.6 A Collision Map

from point A_2 to point B_2. Note that paths of A_1B_1 and A_2B_2 are generally not on a common plane.

Since robot 2 moves on the straight line path A_2B_2, a potential collision between the robots may occur if the sphere of the the wrist of robot 1 intersects the sphere of the wrist of robot 2 during their motions. The distance between two trajectory set points on the robot 1 and the robot 2 paths must be greater than $r_1 + r_2$ for the collision avoidance. The portion that must be avoided by robot 2 at time k can be determined from the intersection of the sphere of radius of $r_1 + r_2$, centered at the point on the path of robot 1 at time k, and the original robot 2 path. The intersected length, however, must be computed for all the servo time instants of potential collisions.

With reference to Figure 4.7, point $\mathbf{p}_1(k)$ is a point on the robot 1 path at time k. The equation of the robot 2 path is denoted as:

$$\mathbf{p}_2 = \mathbf{p}_2(k_0) + \lambda \cdot (\mathbf{p}_2(k_f) - \mathbf{p}_2(k_0)). \tag{4.26}$$

Since the potential collisions between the robots occur at time k when the distance between point $\mathbf{p}_1(k)$ on the robot 1 path and the robot 2 path in Eq. (4.26) is less than or equal to $r_1 + r_2$, we solve the following equation:

$$(r_1 + r_2)^2 = \| \mathbf{p}_1(k) - \mathbf{p}_2 \|^2 \tag{4.27}$$

Using Eq. (4.26) for $\mathbf{p}_2$, we have:

$$\begin{aligned}(r_1 + r_2)^2 = \{ \mathbf{p}_1(k) - \mathbf{p}_2(k_0) - \lambda \cdot (\mathbf{p}_2(k_f) - \mathbf{p}_2(k_0)) \}^T \\ \cdot \{ \mathbf{p}_1(k) - \mathbf{p}_2(k_0) - \lambda \cdot (\mathbf{p}_2(k_f) - \mathbf{p}_2(k_0)) \}\end{aligned} \tag{4.28}$$

More explicitly,

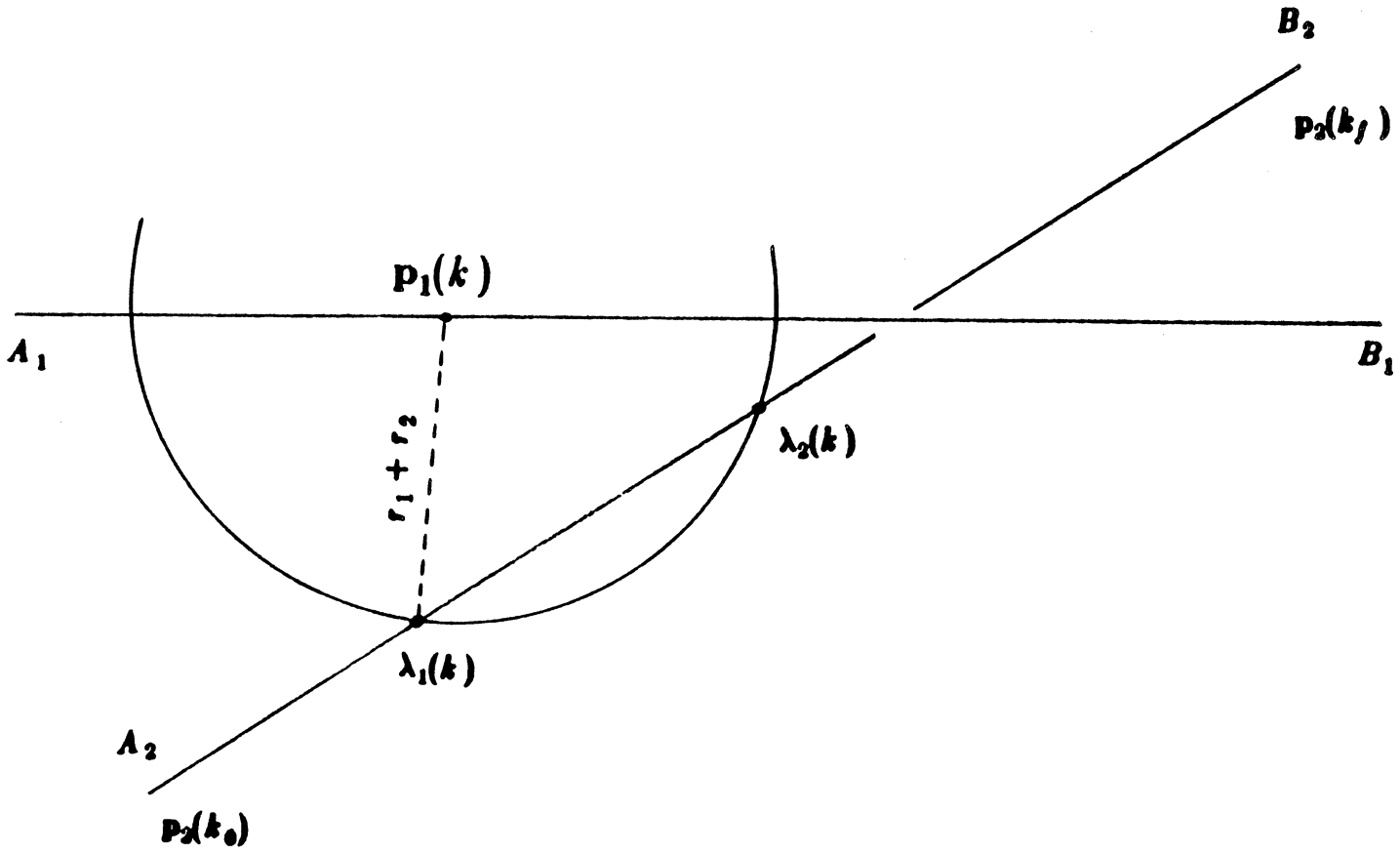


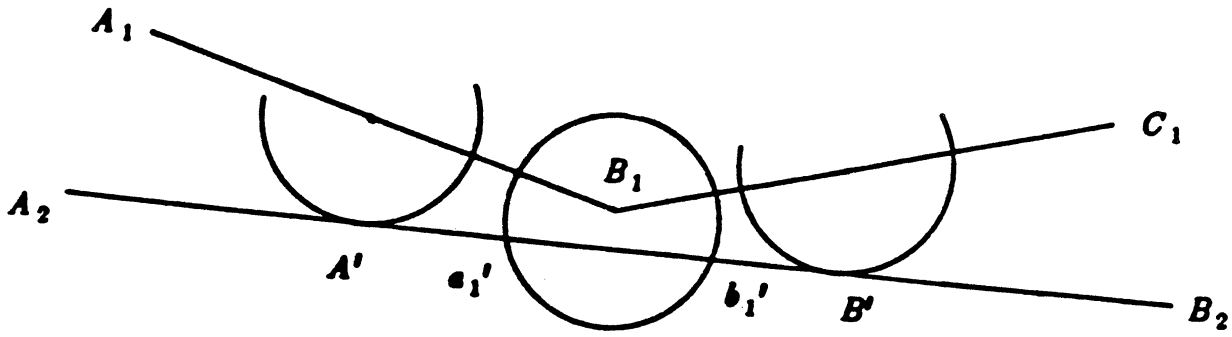
Figure 4.7 Two Robot Path

$$(r_1 + r_2)^2 = \| \mathbf{p}_1(k) - \mathbf{p}_2(k_0) \|^2 - 2\lambda \cdot (\mathbf{p}_1(k) - \mathbf{p}_2(k_0))^T (\mathbf{p}_1(k) - \mathbf{p}_2(k_0)) + \lambda^2 \cdot \| (\mathbf{p}_2(k_f) - \mathbf{p}_2(k_0)) \|^2 \quad (4.29)$$

Eq. (4.29) is a second order equation for λ, which can be solved easily. There are three possible solution cases: (1) Real roots of λ do not exist, (2) Two real roots $\lambda_1(k)$ and $\lambda_2(k)$ exist ($\lambda_1(k) > \lambda_2(k)$.), and (3) Only one real double root $\lambda_1(k)$ exists. When no real root exists, we know that there is no collision between robot 1 and robot 2 at time k. When two real roots exist, we know that the collision lengths range from $l_f \cdot \lambda_1(k)$ to $l_f \cdot \lambda_2(k)$. When only one real double root exists, it marks the beginning or ending times for the collision region. Since the Cartesian trajectory planner discussed in Chapter 3 yields the trajectory information for robot 1 based on the servo time interval, the collision region in the collision map can be obtained by calculating the intersected lengths of the path A_2B_2 for all the servo time instants of potential collisions. It is worth noting that the same analysis applies if the path of robot 1 is a series of straight line segments.

Several boundary points are interpreted in the collision region in Figure 4.8. The points A', B', a_1' and b_1' in Figure 4.8 correspond to the points A, B, a_1 and b_1 in the collision map respectively. For example, we can interpret that the portion a_1b_1 of the collision region is the intersected length or the intersected portion of the robot 2 path, which must be avoided by robot 2 at time $t = k_i T$.

The collision region is drawn in Figure 4.9 as a shaded region. The curve of traveled length versus servo time for robot 2 cannot enter the region if collision is to be avoided. Since it is difficult to describe the geometric shape of the collision region exactly and analytically, a bounding box or a "window" which encloses the whole collision region is established to "approximate" the collision region as shown



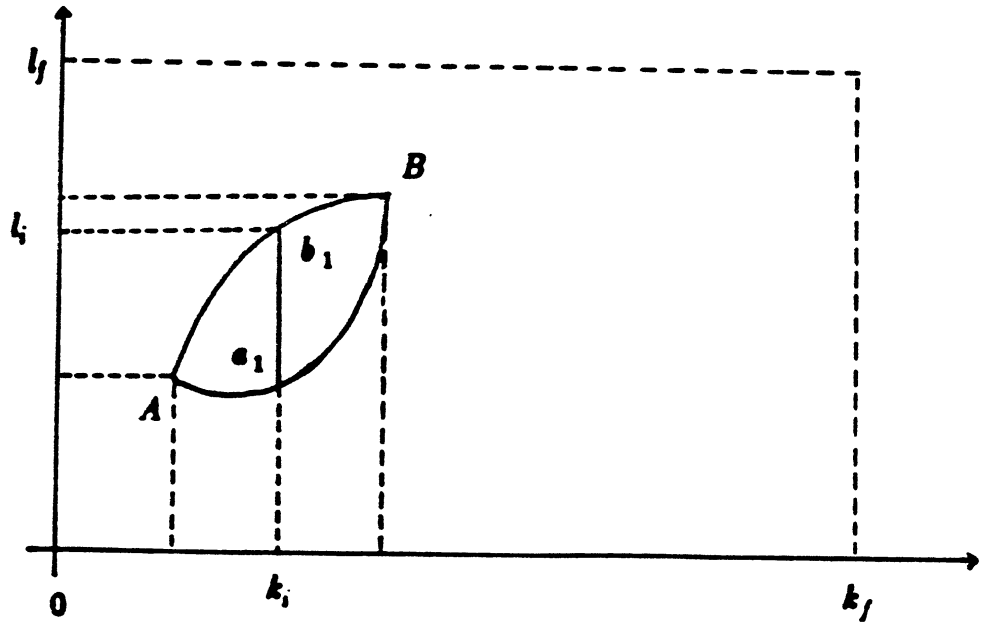


Figure 4.3 Boundary Points for Collision Region

in Figure 4.9. From now on, we will denote the potential collision starting and ending time as k_s and k_e, respectively, and the initial and final time as k_0 and k_f, respectively. Also, we denote the traveled length from point A_2 to the collision starting point and to the collision ending point as l_s and l_e, respectively, and the total traveling length of the path of robot 2 as l_f.

Various collision maps can be identified depending on the collision situations. Figure 4.10 shows the collision maps which could happen in various collision situations. In Figure 4.10-(a), the collision occurs for most of the path of robot 2 for a long time interval and in Figure 4.10-(b), the collision occurs for most of the path of robot 2 for a short time interval. In Figure 4.10-(c), the collision occurs for a small portion of the robot 2 path for a long time interval and in Figure 4.10-(d), the collision occurs for a small portion of the path of robot 2 for a short time interval. If $l_s = 0$, then this corresponds to a special case of Figures from 4.10-(a) to (d). In this case, a different solution to avoid the collision region will be discussed in the next section.

In fact, the traveled length versus servo time curve and the collision region are obtained from the path and trajectory information of two robots, which result from the application of the iterative binary search algorithms in Section 3.3. Thus, the shape of the collision region varies depending on the collision situation. We assumed only a simple collision situation in this section to illustrate the collision map. Multiple collision regions may occur in some collision situations. It is anticipated that a bounding box enclosing the whole scattered collision region may result in rough approximation. However, the discussions presented here can be applied to the case of multiple collision regions also. In the next section, we will utilize the col-

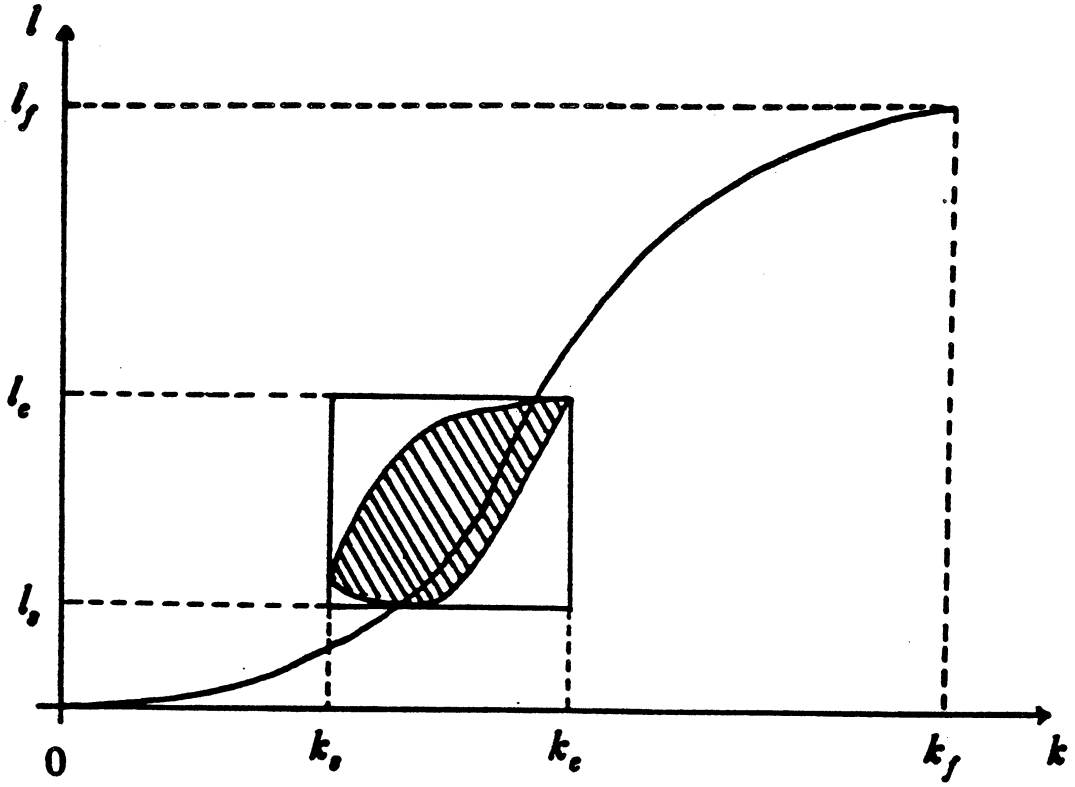


Figure 4.9 Collision Map

lision map to analyze and design the collision-free trajectories for two moving robots.

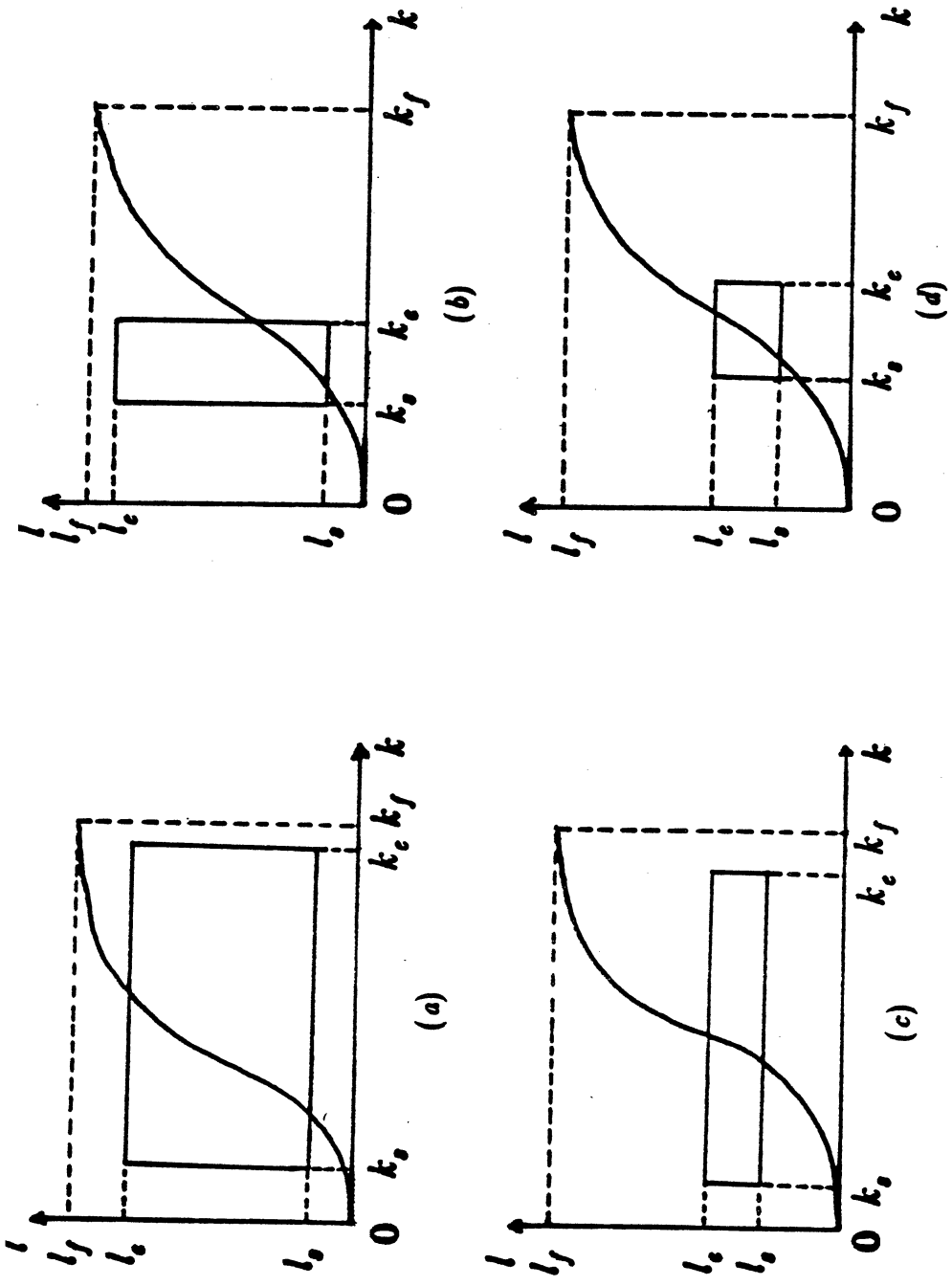


Figure 4.10 Several Types of Collision Region

4.4. Collision-Free Path for Two Moving Robots

Using the collision map and the time scheduling concept, this section presents a method to realize a collision-free motion planning of two moving robots. The collision situations of case 1 and case 2 are discussed separately.

4.4.1. Collision-Free Path Planning for Case 1

In case 1, one robot (robot 2) cannot change its original path but can change its arrival time k_f for collision avoidance with the other moving robot (robot 1). The collision map is utilized in obtaining the collision-free time scheduling of the straight line trajectory.

From the collision map of a collision situation as shown in Figure 4.9, the final arrival time k_f of robot 2 can be altered for the purpose of collision avoidance with robot 1. Since the potential collision starts at time k_s from the collision map, the time for robot 2 to travel the length l_s must be modified so that robot 2 reaches the location of length l_s at a different time. Thus, to find the collision-free time scheduling of the trajectory, we need to find a traveled length versus servo time curve of robot 2 which does not cross or touch the collision region in the collision map.

In Figure 4.11, it is assumed that the traveled length versus servo time curve OM crosses the collision box, which indicates a potential collision. Since the objective is to find a curve like OM' which avoids the collision region, it is necessary to make robot 2 slow down or delay its motion. The arising problems in time scheduling of the original trajectory are then identified as follows:

- Is either a time delay or speed reduction of the robot 2 motion necessary

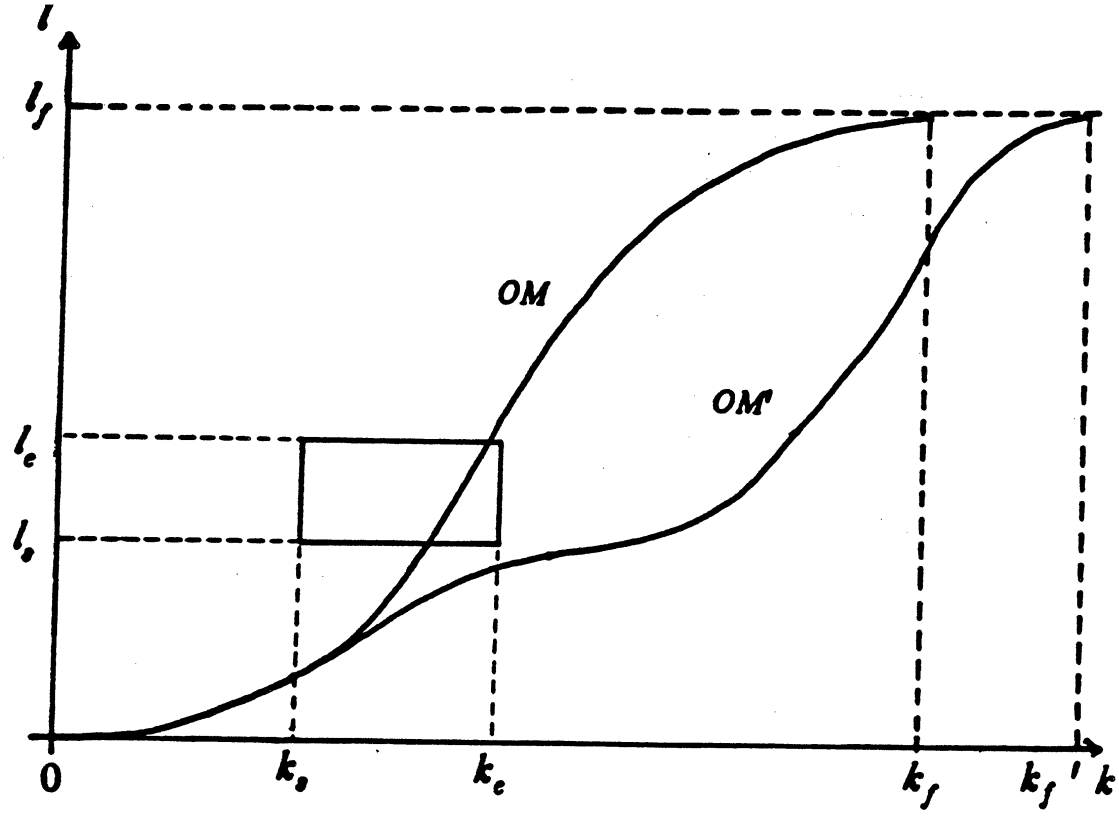


Figure 4.11 A Collision Map with Potential Collisions

for avoiding potential collisions?

- How much must the robot 2 motion be delayed and where must the time delay must be incurred?
- How much must the speed of robot 2 be reduced and where should it occur?
- What are the effects of the time delay and speed reduction on the final arrival time?

These problems can be addressed easily by using the collision map from the collision situation. We assume that a collision map with potential collision between two robots is given, in which the traveled length versus servo time curve of robot 2 crosses the collision region. Depending on the task scheduling requirements and the environment of collision situation, various approaches are discussed in obtaining the traveled length versus servo time curve of robot 2 which can avoid the collision region.

In Figure 4.12, it is assumed that the collision region starts at the location of length l_s at time k_s and ends at the location of length l_e at time k_e, respectively. Also, we denote the final arrival time of robot 2 as k_f from the original trajectory information. Figure 4.12-(a) shows the traveled length versus servo time curve OM_1 which can avoid the collision region by purely delaying the starting time at the initial location. Figure 4.12-(b) shows the traveled length versus servo time curve OM_2 which can avoid the collision region by delaying the starting time at the initial location and reducing the speed of the robot 2 motion. Figure 4.12-(c) shows the traveled length versus servo time curve OM_3 which can avoid the collision region by delaying the robot 2 motion at the location of length l_s and reducing the speed of

the robot 2 motion. Figure 4.12-(d) shows the traveled length versus servo time curves OM_4 and OM_5 which can avoid the collision region by reducing the speed of the robot 2 motion.

In the above discussion, we assumed that the collision region is such that $l_s \neq 0$. If $l_s = 0$, then the only possible solution to avoid the collision region is to delay the robot 2 motion so that robot 2 starts to move after any potential collisions are cleared out on its path.

In Figure 4.12-(a), the curve OM_1 is a curve which avoids the collision region and can be obtained by purely delaying the starting time at the initial location. The time, when robot 2 arrives at the location of length l_s from the original trajectory information, is denoted as k_1. Then, the required time delay at the initial location will be $k_e - k_1$. The arrival time k_f^1 of robot 2 for the curve OM_1 will be $k_f + k_e - k_1$. However, if robot 2 has any constraints to leave the initial location as scheduled, then this curve OM_1 cannot be used for collision avoidance with robot 1 and a different method, which utilizes speed reduction of the robot 2 motion, will be required.

In Figure 4.12-(b), the curve OM_2 is a curve which avoids the collision region and can be obtained by slowing down the robot 2 motion and delaying the starting time at the initial location. The speed reduction of robot 2 is performed on the path of length l_s from the initial location. The time, when robot 2 arrives at the location of length l_s after speed reduction, is denoted as k_2. If we allow the robot to accelerate and decelerate again, the corresponding traveled length versus servo time curve for the segment of length $l_f - l_s$ may touch or cross the collision region again. In this case, the time delay strategy can be realized at the initial location to

avoid the collision region. The required time delay at the initial location will be $k_e - k_2$. The arrival time k_f^2 of robot 2 for the curve OM_2 will be $k_f' + k_e - k_2$, where k_f' is the arrival time of robot 2 for the curve OM_2' only after speed reduction. If $k_e < k_2$, then the speed reduction on the path of length l_s is sufficient to avoid the collision region.

In Figure 4.12-(c), the curve OM_3 is a curve which avoids the collision region and can be obtained by slowing down the robot 2 motion and stopping the movement temporarily at the location of length l_s to cause the required time delay. The time, when robot 2 arrives at the location of length l_s after speed reduction, is denoted as k_3. Then, the required time delay at the location of length l_s will be $k_e - k_3$. The arrival time k_f^3 of robot 2 for the curve OM_3 will be $k_f' + k_e - k_3$, where k_f' is the arrival time of robot 2 for the curve OM_3' only after speed reduction. Again, if $k_e < k_3$, then the time delay at the location of length l_s is not necessary, which corresponds to the case that pure speed reduction is sufficient to avoid the collision region.

In Figure 4.12-(d), the curve OM_4 is a curve which avoids the collision region and can be obtained by slowing down the robot 2 motion adequately. There are a number of ways to reduce the robot 2 speed so that the curve OM_4 can avoid the collision region. Since the curve must avoid the corner point X, speed reduction will be applied to the robot 2 motion on the path of length l_s as an approach to the solution. The curve from the speed reduction of the robot 2 motion can be obtained by applying the time scheduling concept discussed in Section 4.3.2. Since it is assumed that the traveled length versus servo time curve OM exists for the given straight line path, the speed reduction can be realized only on the acceleration por-

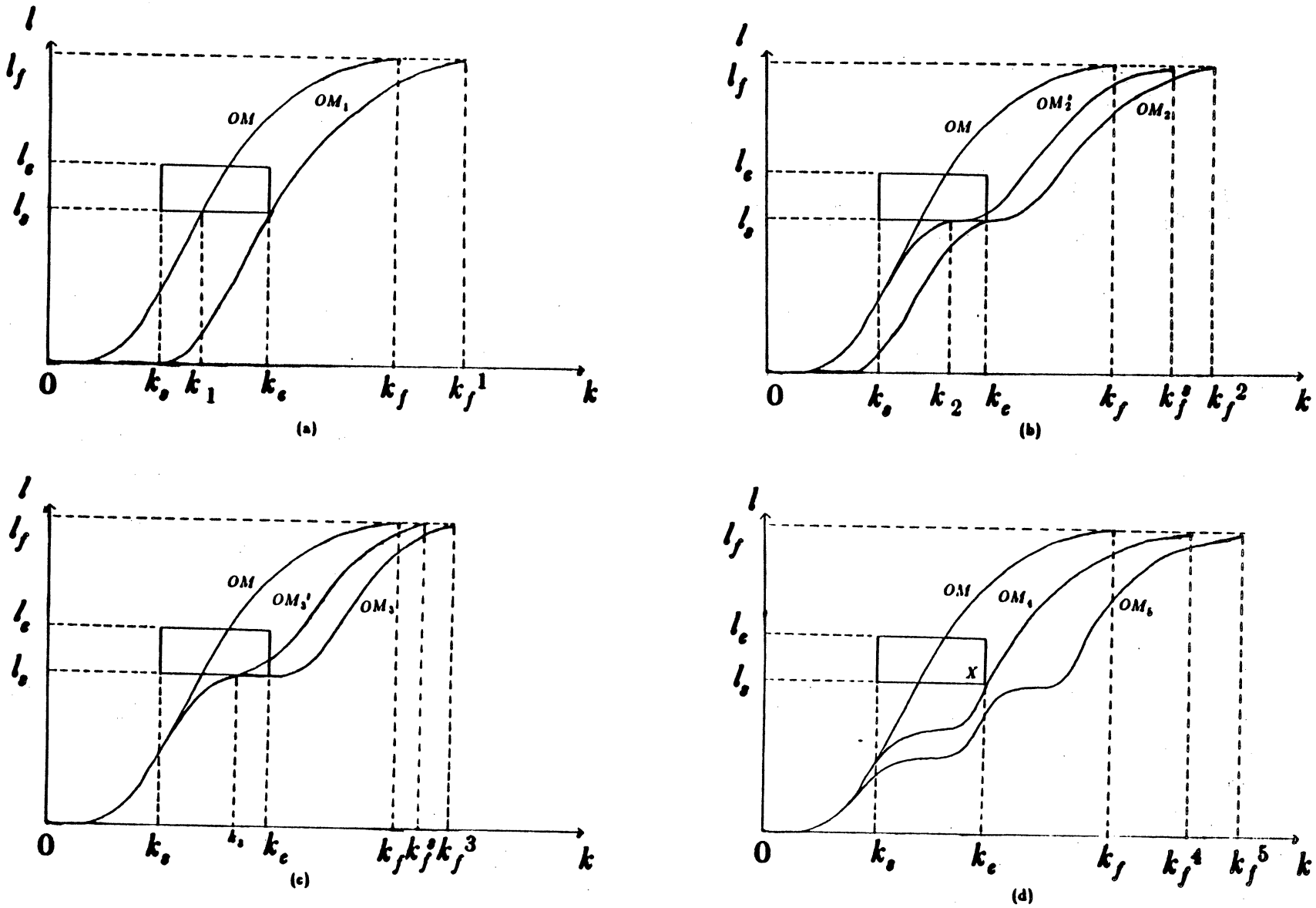


Figure 4.12 Collision-Free Trajectory Curves

tion of the straight line trajectory of robot 2 by selecting an appropriate break point. This corresponds to the explanation that point D is selected on the acceleration portion $\overline{AB}$ in Figure 4.5 to realize the speed reduction.

If one is only interested in pure speed reduction, the speed reduction will be performed initially on the segments of length l_s and $l_f - l_s$. The traveled length versus servo time curve from planning the two segments may still touch or cross the collision region. Then, further speed reduction must be applied to the segment of length l_s. This can be accomplished by dividing this segment of length l_s into two, three, or four, etc. segments so that each segment can be planned with a break point until the combined traveled length versus servo time curve does not touch or cross the collision region. As an example, the curve OM_5 is drawn from the combined trajectory information for three segments of length $\frac{l_s}{2}$, $\frac{l_s}{2}$, and $l_f - l_s$ in Figure 4.12-(d). Recall that if we use more break points in the trajectory planning of a path, the total traveling time will be much longer. Thus the final arrival time for the curve OM_5 will be much longer than that of the curve OM.

The effects on the final arrival time due to the time scheduling (i.e. time delay and/or speed reduction) of a trajectory are considered in more detail. Hereafter, discussions are based on the assumption that the servo time interval is very small and the effects from discretization are negligible. Later, the results will be verified from a computer simulation.

We consider the collision situations shown in Figure 4.13, where the curve OM_O is a traveled length versus servo time curve from the original trajectory information of robot 2 and the curve OM is a time scheduled traveled length versus servo time curve with a final arrival time k_f. The curve OM can be obtained by

reducing the robot 2 speed on the path of length l_s from the initial location. The time k_1 is the time when the original curve OM_O reaches the location of length l_s. The time k_2 is the time when the curve OM reaches the location of length l_s after speed reduction.

If the robot is allowed to delay its starting time by $k_e - k_1$, then the original curve OM_O is shifted to avoid the collision region in the collision map, which corresponds to the curve OM_1 in Figure 4.13. By the speed reduction of the robot 2 motion, a curve OM can be obtained also. Here, it is possible that $k_2 \geq k_e$, or $k_2 < k_e$ to occur in Figure 4.13. When $k_2 \geq k_e$, the speed reduction on the path of length l_s will be sufficient to obtain a curve for avoiding the collision region. When $k_2 < k_e$, speed reduction is not sufficient for collision avoidance and pure time delay at an appropriate location can be used for collision avoidance, which corresponds to the curve OM_2' in Figure 4.12-(b). In Figure 4.12-(b), the curve OM_2 shows a time delay at the initial location with speed reduction, while in Figure 4.12-(c), the curve OM_3 shows a time delay at an intermediate location with speed reduction to avoid the collision region.

If we assume that time delay is not allowed for the robot 2 motion, then k_2 must be equal to or greater than k_e for avoiding the collision region. Also, for the speed reduction on the acceleration portion, we have $k_1 \leq k_2$. That is, the average speed of the robot for the curve OM is equal to or smaller than that for the curve OM_O on the path of length l_s, which results in the fact that the traveling time corresponding to the curve OM is longer than that to the curve OM_O. Thus the following inequality from speed reduction must be satisfied for the collision avoidance with robot 1.

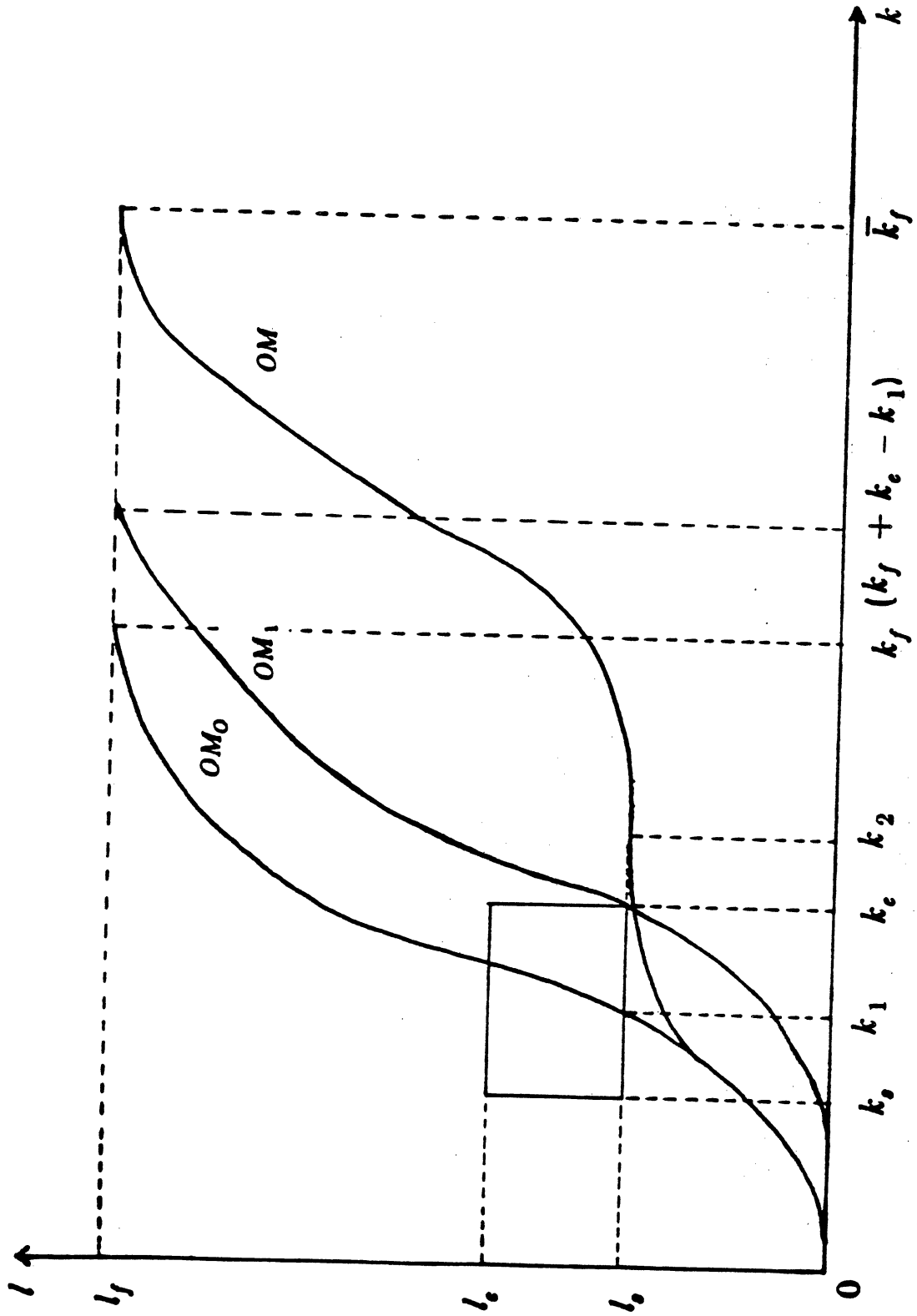


Figure 4.13 Collision Map after Time Scheduling

$$k_1 \leq k_e \leq k_2 \tag{4.30}$$

The curve OM_O is obtained from the trajectory information constituting an acceleration portion and a deceleration portion of the straight line trajectory. It is notable that the traveling time $k_f - k_1$ is the lower bound for all those motions going from the location of length l_e to that of length l_f, where the velocity at l_e is less than the velocity of OM_O at l_e. Thus, considering the traveling time on the path of length $l_f - l_e$ when the robot follows the curve OM (where the velocity at k_2 is equal to zero.), we have:

$$k_f - k_1 < \bar{k}_f - k_2 \tag{4.31}$$

or

$$\bar{k}_f > k_f + k_2 - k_1 \tag{4.32}$$

where time $\bar{k}_f$ is the final arrival time of curve OM.

When the speed reduction is performed on the segment of length l_e for avoiding the collision box, we want to show that the final arrival time from the speed reduction is longer than that from the time delay of the robot 2 motion. Since $k_e - k_1 \leq k_2 - k_1$, we have:

$$\bar{k}_f > k_f + k_2 - k_1 \geq k_f + k_e - k_1 \tag{4.33}$$

Therefore,

$$\bar{k}_f > k_f + k_e - k_1 \tag{4.34}$$

The time $k_e - k_1$ is the required time delay for the curve to avoid the collision region (the curve OM_1 in Figure 4.13). The above equation indicates that the final arrival time from the speed reduction on the path of length l_e is greater than that from the time delay. However, if speed reduction occurs at an arbitrary location of

the straight line path, the final arrival time after time delay may not always give rise to shorter traveling time than that after speed reduction of the robot motion. These aspects will be discussed with a simulation in Section 6.4.

A procedure is outlined below for the time scheduling of a straight line trajectory. From the discussion of Section 4.3.2, a segmentation process for the path of collision starting length l_s is described for speed reduction instead of specifying the locations of the break points. The procedure utilize a user-designated integer δ_4 for specifying the maximum number of the segments on the path of length l_s. If a potential collision still occurs even after the path is segmented by δ_4, then time delay of the robot 2 motion will be used for the purpose of collision avoidance.

Procedure TS (Time Scheduling): This procedure addresses the iterative steps in obtaining the time scheduling of the straight line trajectory for the collision avoidance of two moving robots.

TS1. [Initialize the index i.] Set $i = 1$.

TS2. [Construct a collision map from the collision situation.] Obtain a collision map from the path and trajectory information of robots 1 and 2.

TS3. [Obtain the traveled length versus servo time curve.] Identify the collision starting length l_s. Then apply the algorithm FW, the procedure BR, and the algorithm MFW to the segments of length l_s and $l_f - l_s$ and draw the combined traveled length versus servo time curve from the resultant trajectory information in the collision map of step TS2.

TS4. [Check the collision map and divide the segment of length l_s if necessary.]
If the curve in step TS3 still touches the collision box,
then increase i by 1, and divide the segment of length l_s by i, and go to step TS5.
otherwise, stop. (Planning of collision-free path by time scheduling of the trajectory is successful.)

TS5. [Obtain the combined traveled length versus servo time curve.] Apply the search algorithm FW, the procedure BR, and the algorithm MFW to each of the segments, draw the combined traveled length versus servo time curve in the collision map, and go to step TS6.

TS6. [Check the stopping variable.]

If $i \geq \delta_4$, where δ_4 is a designated integer that specifies the maximum allowable number of segments on the path of length l_s,
then time delay of the robot 2 motion will be used for the collision avoidance purpose.
otherwise, go to step TS4.

4.4.2. Collision-Free Path Planning for Case 2

A collision-free path of robot 2 is considered in the following categories: (1) when time scheduling is applied without any path modification, (2) when only path modification is applied without considering the trajectory information of two robots, and (3) when path modification and time scheduling are applied simultaneously.

Since the arrival time can be changed in case 2, the time scheduling on the original robot 2 path, which is similar to the solution in case 1, is considered first. As indicated in Eq. (4.34), if speed reduction is performed on the path of length l_s, time delay yields the smaller final arrival time than speed reduction for avoiding a potential collision. Thus, time delay is considered in this section for the time scheduling of a trajectory. If we denote the final arrival time of robot 2 as k_f for the original trajectory, then the total traveling time after a time delay will be:

$$k_{TS} = k_f + \Delta k_f \tag{4.35}$$

where Δk_f is a required time delay for avoiding a potential collision. If Δk_f is very small compared to k_f, then time delay of the original trajectory provides us a good solution for the purpose of collision avoidance.

When Δk_f is fairly large and $l_s = 0$ (See Figure 4.15), we can consider a solution in categories (2) and (3). If a path deviates from the robot 1 path for at least a distance of $r_1 + r_2$, then the path can be a collision-free path and constitute a solu-

tion in category (2). It is notable that the total traveling time of this path can be smaller, equal to, or larger than k_{TS} in Eq. (4.35). The deviation of this path can be reduced by considering a solution in category (3), where path modification and time delay are used at the same time. If path modification and time delay are applied simultaneously for planning a collision-free path, there exists an infinite number of collision-free paths of the two robots. Since two robots are working simultaneously in a common workspace, and their movements are coordinated in a time sequence, there is likely to be a constraint on the time delay of the robot 2 motion. Also, it is desirable to plan a collision-free path with the smallest deviation from the original path. Thus, planning of a collision-free path is considered with respect to an allowable time delay at the initial location and the minimum deviation.

First, we find a collision-free path which deviates from the robot 1 path for at least a distance of $r_1 + r_2$ through a geometric analysis. The path is then guaranteed for the collision avoidance with robot 1. A collision situation is shown in Figure 4.14, where robot 1 moves from A_1 to B_1, while robot 2 moves from A_2 to B_2. We assume that a potential collision exists between the two moving robots with a corresponding collision map. See Figure 4.15 for the collision map, where P and Q are the corner points of the collision region at which the potential collision is assumed to start at time k_s and end at time k_e, respectively. The length $l_{A_2B_2}$ is the total length of the path from A_2 to B_2. The following analysis indicates how to choose a point C_2 such that the path from A_2 to B_2 via C_2 deviates from the robot 1 path for at least a distance of $r_1 + r_2$.

With reference to Figure 4.14, the distance between the robot 1 path and the robot 2 path and corresponding nearest points on each path will be found. The

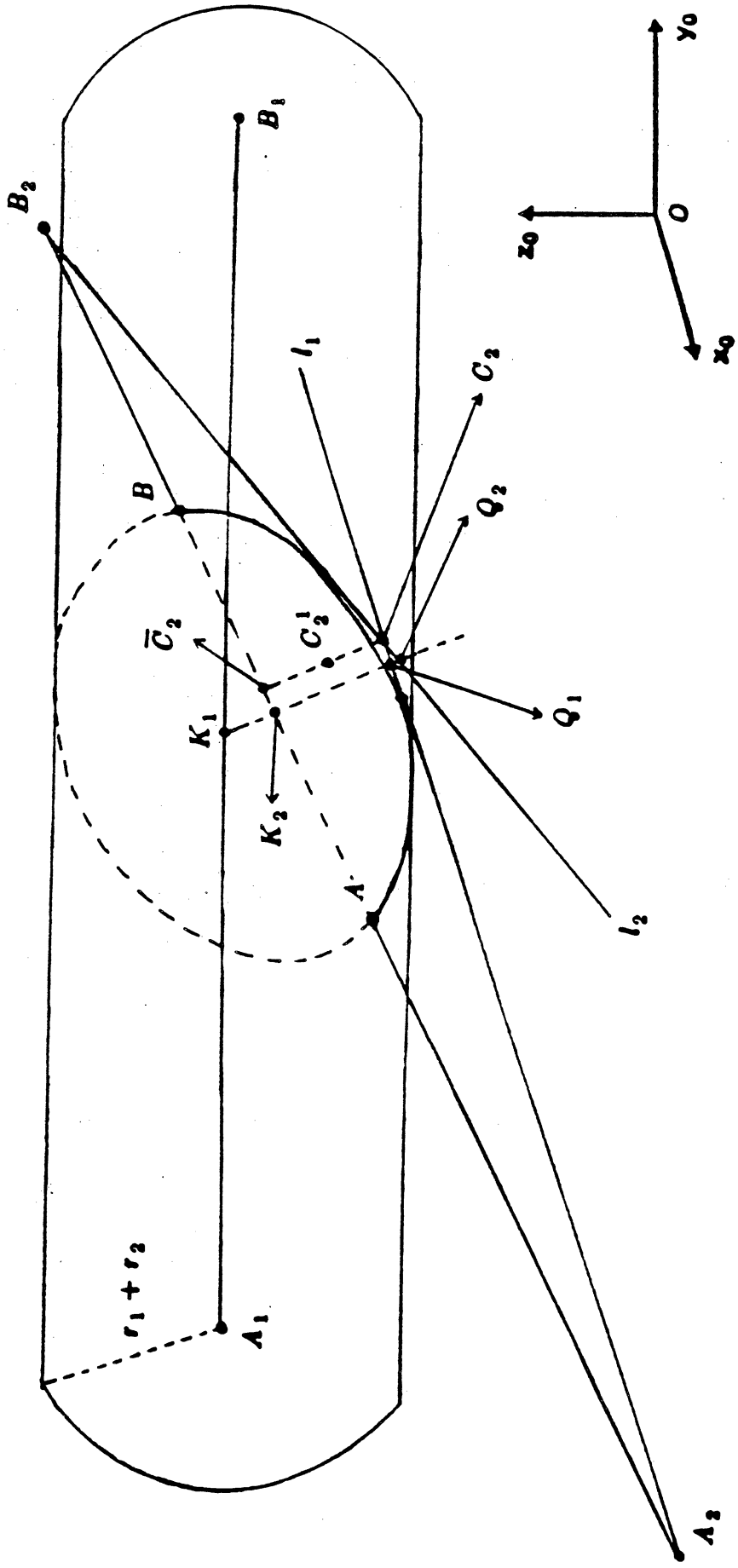


Figure 4.14 Collision Avoidance by Path Modification

straight line path A_1B_1 is a line which passes through a point $A_1(x_{A1}, y_{A1}, z_{A1})$ with the direction number of $(x_{B1} - x_{A1}, y_{B1} - y_{A1}, z_{B1} - z_{A1})$. Thus, the general form of an arbitrary point G_{A1B1} on the path A_1B_1 can be represented by a vector notation as:

$$\vec{OG_{A1B1}} = \{ \lambda_1(x_{B1} - x_{A1}) + x_{A1}, \lambda_1(y_{B1} - y_{A1}) + y_{A1}, \lambda_1(z_{B1} - z_{A1}) + z_{A1} \} \tag{4.36}$$

where λ_1 is a parameter for representing the straight line path A_1B_1. Similarly, the general form of an arbitrary point G_{A2B2} on the path A_2B_2 can be represented by a vector notation as:

$$\vec{OG_{A2B2}} = \{ \lambda_2(x_{B2} - x_{A2}) + x_{A2}, \lambda_2(y_{B2} - y_{A2}) + y_{A2}, \lambda_2(z_{B2} - z_{A2}) + z_{A2} \} \tag{4.37}$$

where λ_2 is a parameter for representing the straight line path A_2B_2. Let us denote the distance between G_{A1B1} and G_{A2B2} as DS_1. Then the square of the distance DS_1 can be obtained as:

$$DS_1^2 = \| \vec{OG_{A1B1}} - \vec{OG_{A2B2}} \|^2 \tag{4.38}$$

To find the nearest two points on each path, the partial derivatives of DS_1^2 with respect to λ_1 and λ_2 are obtained. Since the partial derivatives are zeros at the nearest two points of the two straight line paths, we have:

$$\frac{\partial DS_1^2}{\partial \lambda_1} = 0 \quad ; \quad \frac{\partial DS_1^2}{\partial \lambda_2} = 0 \tag{4.39}$$

Solving Eq. (4.39) simultaneously, we can find λ_1 and λ_2, which correspond to the nearest two points between the two straight line paths. We denote these two points on the robot 1 and robot 2 path as $K_1(x_{K1}, y_{K1}, z_{K1})$ and $K_2(x_{K2}, y_{K2}, z_{K2})$, respectively, as shown in Figure 4.14. It is notable that points A_2, B_2, K_1, and K_2 are on the same plane which can be constructed by $\overline{A_2B_2}$ and $\overline{K_1K_2}$.

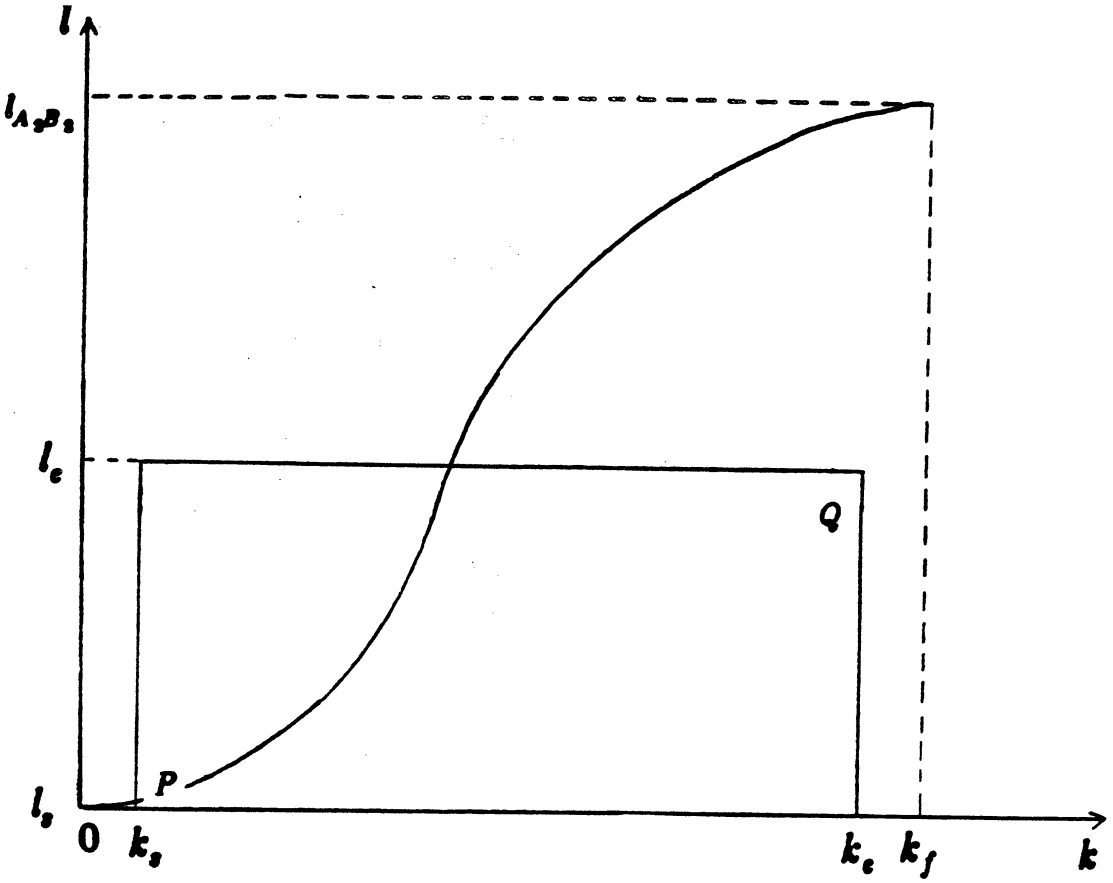


Figure 4.15 Initial Collision Map

It is notable that the swept volume of a sphere in the direction of a straight line forms a cylinder capped with two semi-spheres at both ends. Thus, for considering the distance of $r_1 + r_2$ between the two robots, we view the collision situation between the cylinder surface of radius $r_1 + r_2$ in the direction of A_1B_1 and the robot 2 path. There are two intersection points on the cylinder surface, we call them A and B, at which the cylinder surface is perforated by the robot 2 path. To maintain the distance of $r_1 + r_2$ between the robots, we consider the intersecting curve between the cylinder surface and the plane which is constructed by $\overline{A_2B_2}$ and $\overline{K_1K_2}$. We call the intersecting curve on the cylinder surface a collision-free locus. The collision-free locus connecting A and B forms a part of an ellipse, which is centered at K_1 (See Figure 4.14). Since K_1 and K_2 are the nearest two points between the two straight line paths, it is easy to show that axes of the ellipse are in the directions of $\overline{A_2B_2}$ and $\overline{K_1K_2}$. Thus, the maximum deviation of the collision-free locus from K_1 occurs in the direction of $\overline{K_1K_2}$.

In order to obtain straight line segments which approximate the collision-free locus, we construct lines l_1 and l_2 which lead separately from A_2 and B_2 and are tangent to the ellipse. Consider two points Q_1 and Q_2 in the direction of $\overline{K_1K_2}$. We denote their deviations from K_1 as d^1 and d^2, respectively. First, we obtain the location of Q_1 and Q_2 in terms of d^1 and d^2, respectively. Then, we find constraints on d^1 and d^2 to guarantee that the distances between two straight line paths A_1K_1 and A_2Q_1, and B_1K_1 and B_2Q_2 are equal to $r_1 + r_2$.

The location of $Q_1(x_{Q1}, y_{Q1}, z_{Q1})$ is obtained in terms of d^1 as:

$$\vec{OQ}_1 = \vec{OK}_1 + \frac{\vec{OK}_2 - \vec{OK}_1}{\| \vec{OK}_2 - \vec{OK}_1 \|} \cdot d^1 \qquad (4.40)$$

Since the value of d_1 specifies the location of Q_1, we can represent the distance between two straight line paths A_1K_1 and A_2Q_1 in terms of the deviation d^1 [6]. If we denote this distance as DS_2, then:

$$DS_2 = f_1(d^1) \tag{4.41}$$

where f_1 relates the distance between the paths of A_1K_1 and A_2Q_1 with the deviation d^1. To maintain the distance between the paths A_1K_1 and A_2Q_1 for at least $r_1 + r_2$, we must have:

$$DS_2 = f_1(d^1) \geq r_1 + r_2 \tag{4.42}$$

Here, we will consider the equality only to avoid an unnecessary large deviation from K_1. Then, we can obtain a constraint on the deviation d^1 as:

$$d^1 = f_1^{-1}(r_1 + r_2) \tag{4.43}$$

Similarly, for the paths of B_1K_1 and B_2Q_2, we have a constraint on the deviation d^2 as:

$$d^2 = f_2^{-1}(r_1 + r_2) \tag{4.44}$$

where f_2 relates the distance between the paths of B_1K_1 and B_2Q_2 with the deviation d^2. The location of Q_1 can be obtained from Eq. (4.40) by using d^1 of Eq. (4.43). Also, the location of Q_2 can be obtained from Eq. (4.40) by substituting d^2 of Eq. (4.44) for d^1.

Using the locations of Q_1 and Q_2, the two straight lines l_1 and l_2 in the directions of $\overline{A_2Q_1}$ and $\overline{B_2Q_2}$ (See Figure 4.14.) can be identified easily. It is notable that l_1 and l_2 are on the same plane which is constructed by $\overline{A_2B_2}$ and $\overline{K_1K_2}$. Now, we consider the intersection point of these two straight lines l_1 and l_2.

Apparently, the path from A_2 to B_2 via the intersection point, which is denoted as C_2, is a collision-free path in category (2). To find the deviation of this path from the original robot 2 path, the nearest point on $\overline{A_2B_2}$ from C_2, which we call $\bar{C}_2$, is found from a vector projection and addition as:

$$\vec{OC_2} = \vec{OB_2} + (\vec{OA_2} - \vec{OB_2}) \cdot \frac{\vec{B_2C_2} \cdot (\vec{OA_2} - \vec{OB_2})}{|| \vec{OA_2} - \vec{OB_2} ||^2} \tag{4.45}$$

Then the actual deviation of the collision-free path from the original robot 2 path is found as:

$$d^{act} = || \vec{OC_2} - \vec{O\bar{C}_2} || \tag{4.46}$$

Note that no time delay of the robot 2 motion is required for the path from A_2 to B_2 via C_2 for the purpose of collision avoidance, while Δk_f in Eq. (4.35) is required for the original robot 2 path. Of course the time of travel from A_2 to B_2 via C_2 will exceed K_f.

We now consider a collision-free path by both path modification and time delay. A path, which deviates from $\bar{C}_2$ for a distance of Δd, is considered. A point C_2^1 can be found for the deviation of Δd from $\bar{C}_2$ as:

$$\vec{OC_2^1} = \vec{O\bar{C}_2} + \frac{\vec{OC_2} - \vec{O\bar{C}_2}}{|| \vec{OC_2} - \vec{O\bar{C}_2} ||} \cdot \Delta d \tag{4.47}$$

The point C_2^1 specifies the lengths of $\overline{A_2C_2^1}$ and $\overline{C_2^1B_2}$ as $\bar{l}_1^1$ and $\bar{l}_2^1$, respectively, with the corresponding collision map as shown in Figure 4.16, where the traveled length versus servo time curve OM^1 is assumed to cross the collision region for the path $A_2 \longrightarrow C_2^1 \longrightarrow B_2$. The required time delay for avoiding the collision region can be found from the trajectory information. The traveled length versus

servo time curve OM^2 is a curve, which is obtained from OM^1 with the required time delay for the path $A_2 \longrightarrow C_2^1 \longrightarrow B_2$.

As mentioned earlier, the two robots are moving simultaneously in a time-coordinated sequence. Other objects may need to move close to the initial location of robot 2. Thus, there is likely to be a constraint on the time delay at the initial location for avoiding the potential collision. If the allowable time delay at the initial location is denoted as Δk_{allow}, we want to obtain a collision-free path which does not violate this constraint. It is notable that the path from A_2 to B_2 via C_2 does not need any time delay for the purpose of collision avoidance. Thus, this path always meets the constraint on the time delay at the expense of the deviation d^{act}. If Δk_f in Eq. (4.35) is smaller than or equal to Δk_{allow}, then the solution in category (1) will be enough for the purpose of collision avoidance. However, if Δk_f is bigger than Δk_{allow}, then time delay on the original robot 2 path cannot be used for the purpose of collision avoidance. Recall that speed reduction on the original robot 2 path is not appropriate since we are considering the case that $l_s = 0$ (See Figure 4.15). In this aspect, we want to find a collision-free path by both path modification and time delay of the robot 2 motion. Note, however, that although the time delay we determine will be less than Δk_{allow}, the total traveling time may be larger than k_{TS}.

If the required time delay for a path between $\overline{A_2B_2}$ and the path from A_2 to B_2 via C_2^1 exceeds Δk_{allow}, then we can increase the path deviation from the original robot 2 path. Otherwise, we can decrease the path deviation from the original robot 2 path. A bisection method between $\overline{C}_2$ and C_2 is suggested to increase or to decrease the path deviation depending on whether the required time delay exceeds

the allowable time delay Δk_{allow} or not. Using the bisection method, we develop an iterative procedure for obtaining a collision-free path of robot 2 subject to the performance measure (2) (Refer Section 4.2.1) and a constraint on the time delay(Δk_{allow}). Note that, since the path from A_2 to B_2 via C_2 does not need any time delay for the purpose of collision avoidance, the existence of a collision-free path from the bisection process is always guaranteed.

A procedure is outlined next for planning a collision-free path in category (3) subject to the performance measure (2) and a constraint on the time delay of the robot 2 motion.

Procedure PM (Path Modification): This procedure addresses the iterative steps for obtaining a collision-free path in category (3) subject to the performance measure (2) and a constraint on the time delay of the robot motion. The δ_5 in step PM7 specifies the maximum iteration number of the bisection process for stopping the procedure. This procedure is referred to the collision situation shown in Figure 4.14 and the collision map in Figure 4.15.

PM1. [Initialize the index i.] Initialize $i = 1$.

PM2. [Compute the necessary information.] Obtain C_2 (Eqs. (4.36)-(4.44)), $\overline{C}_2$ (Eq. (4.45)), and d^{act} (Eq. (4.46)), set $\Delta d(1) = \frac{d^{act}}{2}$, and obtain C_2^1 corresponding to $\Delta d(1)$ (Eq. (4.47)).

PM3. [Draw the collision map.] Obtain the collision map corresponding to the robot 2 path from A_2 to B_2 via C_2^i.
If a potential collision exists,
then go to step PM4.
otherwise go to step PM6.

PM4. [Compute the required time delay.] Using the collision map and trajectory information, obtain the required time delay for the path from A_2 to B_2 via C_2^i for the purpose of collision avoidance.

PM5. [Increase the deviation from the original path.]
If the required time delay on the path from A_2 to C_2^i exceeds the constraint on time delay,
then compute $\Delta d(i+1)$ such that $\Delta d(i+1) = \Delta d(i) + \frac{d^{act}}{2^{i+1}}$, find C_2^{i+1}

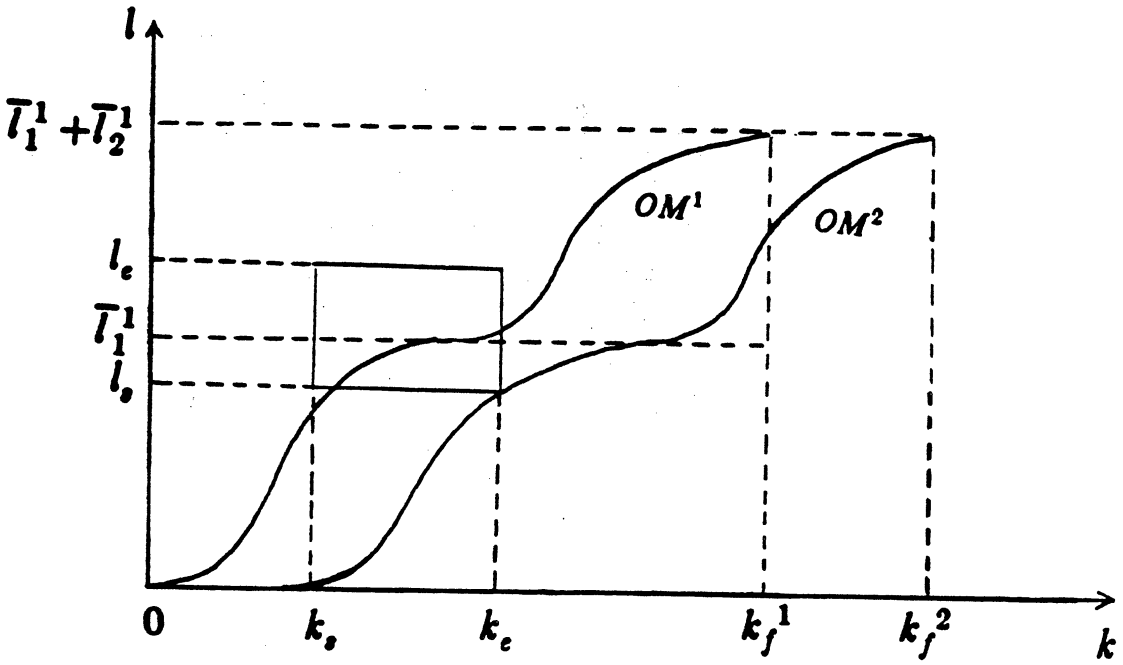


Figure 4.16 Collision Map with Path Modification

corresponding to $\Delta d(i+1)$ (Eq. (4.47)), and go to step PM3 with the updated value of i.

otherwise, go to step PM6.

PM6. [Decrease the deviation from the original path.] Compute $\Delta d(i+1)$ such that $\Delta d(i+1) = \Delta d(i) - \frac{d^{act}}{2^{i+1}}$, find C_2^{i+1} corresponding to $\Delta d(i+1)$ (Eq. (4.47)), and go to step PM7 with the updated value of i.

PM7. [Check the stopping variable.]

If $i \geq \delta_5$, where δ_5 is a user-designated integer that specifies the global stopping of the bisection process,

then stop. (The path from A_2 to B_2 via C_2^i is a collision-free path subject to the performance measure (2) and the constraint on time delay with δ_5 for stopping the bisection process.)

otherwise, go to step PM3.

4.5. Summary

Collision-free path planning problem of two robots is investigated systematically by using a sphere model for the wrist of the robot, straight line path planning, and the notions of collision map and time scheduling. Using geometric analysis, connected straight line segments have been obtained subject to the performance measure to obtain a collision-free path for a moving robot in the presence of a stationary robot. Discussions of the relationship between the number of via points and the allowable deviation error of the collision-free path are also presented. Several collision situations are identified for obtaining collision-free paths of two moving robots. Depending on the collision situation, the time scheduling of the original trajectory and/or path modification have been used to obtain the collision-free paths. The resultant collision-free trajectory will be used in an adaptive perturbation control in Cartesian space, which will be discussed in the next chapter.

CHAPTER 5

CONTROL OF MANIPULATOR IN CARTESIAN SPACE

5.1. Introduction

The purpose of manipulator control is to maintain a prescribed motion for the manipulator along a desired time-based trajectory by applying corrective compensation torques to the actuators to adjust for any deviations of the manipulator from the trajectory. In this Chapter, we present a Cartesian space adaptive perturbation control strategy [28] which controls the manipulator hand to follow a collision-free straight line trajectory as closely as possible under various loading conditions.

In the Cartesian space control, we want small position and velocity errors along the desired hand trajectory. Thus, the control feedback should be performed at the hand level. Also, the changes of the load in a task cycle should be taken into account so that the accuracy of the controller is not significantly effected by them. This suggests an adaptive control approach.

We will utilize the Cartesian trajectory planner developed in Chapter 3, and concentrate on developing a Cartesian space control method which is based on resolved motion and adaptation. The necessary kinematic relationships between the Cartesian coordinates and joint coordinates are presented again in Section 5.2 to develop equations of motion of the manipulator hand. In Section 5.3, the equations

of motion of the manipulator are derived for formulating the adaptive control scheme. The resolved motion adaptive control, which is based on the perturbation theory and parameter identification scheme, is presented in Section 5.4. In Section 5.5, the computational complexity of the resolved motion adaptive control scheme is discussed. Finally, summaries and discussions are presented in Section 5.6

5.2. Kinematics of the Manipulator Hand

The kinematic relationships of position, velocity and acceleration between the joint space and the Cartesian space are required to obtain the equations of motion of the manipulator in the Cartesian space. The location of the manipulator hand with respect to a fixed reference coordinate system can be realized by establishing an orthonormal hand coordinate frame at the gripping point of the hand. Figure 2.7 is shown again in Figure 5.1. Let us define the position, orientation, linear velocity, and angular velocity vectors of the manipulator hand with respect to the reference frame respectively as in Eq. (2.3),

$$\begin{aligned} \mathbf{p}(t) &\stackrel{\Delta}{=} (p_x(t), p_y(t), p_z(t))^T \quad ; \quad \Phi(t) \stackrel{\Delta}{=} (\alpha(t), \beta(t), \gamma(t))^T \\ \mathbf{v}(t) &\stackrel{\Delta}{=} (v_x(t), v_y(t), v_z(t))^T \quad ; \quad \Omega(t) \stackrel{\Delta}{=} (\omega_x(t), \omega_y(t), \omega_z(t))^T \end{aligned} \tag{5.1}$$

where the superscript "T" denotes transpose operation on vectors and matrices. The time derivative of the position of the hand with respect to the reference frame is equal to the linear velocity of the hand, as derived in Eq. (2.10),

$$\dot{\mathbf{p}}(t) = \frac{d\,\mathbf{p}(t)}{dt} = \mathbf{v}(t) \tag{5.2}$$

From Eq. (2.13), the relationship between the $(\dot{\alpha}(t), \dot{\beta}(t), \dot{\gamma}(t))^T$ and $(\omega_x(t), \omega_y(t), \omega_z(t))^T$ is,

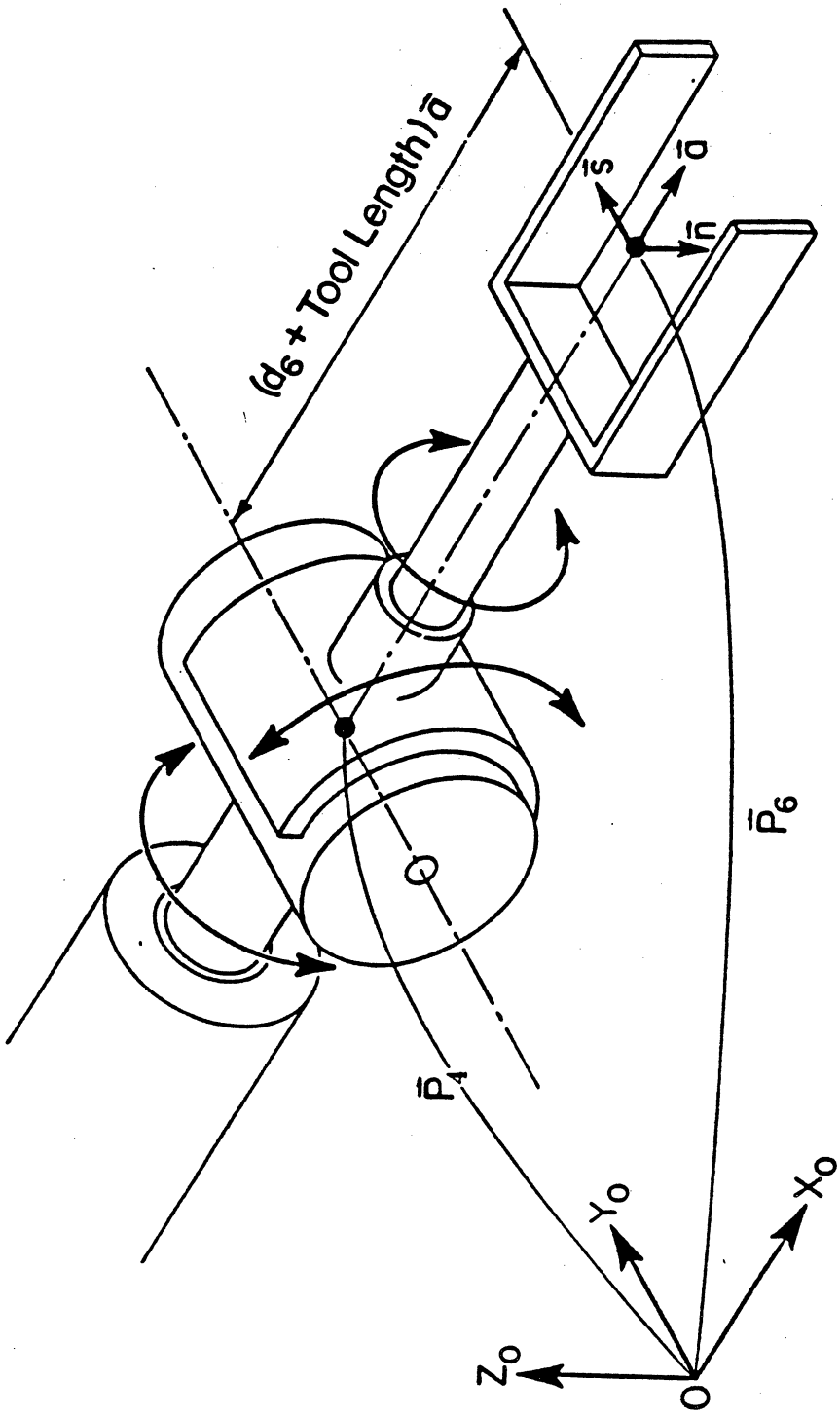


Figure 5.1 Hand Coordinate System

$$\begin{bmatrix} \dot{\alpha}(t) \\ \dot{\beta}(t) \\ \dot{\gamma}(t) \end{bmatrix} = \sec\beta \begin{bmatrix} C\gamma & S\gamma & 0 \\ -S\gamma C\beta & C\gamma C\beta & 0 \\ C\gamma S\beta & S\gamma S\beta & C\beta \end{bmatrix} \begin{bmatrix} \omega_x(t) \\ \omega_y(t) \\ \omega_z(t) \end{bmatrix} \tag{5.3}$$

or

$$\dot{\Phi}(t) \triangleq \left[M(\Phi) \right] \Omega(t) \tag{5.4}$$

The linear and angular acceleration of the manipulator hand can be found from the Eq. (2.16) as,

$$\begin{bmatrix} \dot{v}(t) \\ \dot{\Omega}(t) \end{bmatrix} = [\, \dot{J}(q, \dot{q})\,]\,[\, J^{-1}(q)\,] \begin{bmatrix} v(t) \\ \Omega(t) \end{bmatrix} + [\, J(q)\,]\, \ddot{q}(t) \tag{5.5}$$

The Eqs.(5.2)-(5.5) are the equations required to obtain hand information from joint information.

5.3. Equations of Motion of the Manipulator

To formulate the resolved motion control method, we obtain the manipulator equations of motion by using the formulas developed in Section 5.2.

The dynamics of a six-joint manipulator can be derived either by Lagrange-Euler [5,24,46] or Newton-Euler formulations [24,38]. The resulting equations of motion are highly nonlinear and consist of inertia loading, coupling reaction forces (Coriolis and centrifugal) among the various joints, and gravity loading effects. In general, the Lagrange-Euler equations of motion of a six-joint manipulator, excluding the actuator dynamics, gear friction and backlash, can be expressed in vector matrix notation as,

$$D(q)\,\ddot{q}(t) + h(q, \dot{q}) + c(q) = \tau(t) \tag{5.6}$$

where $\tau(t)$ is a 6×1 applied torque vector for joint actuators, q is the angular positions, $\dot{q}$ is the angular velocities, $\ddot{q}(t)$ is a 6×1 joint acceleration vector, c(q) is a 6×1 gravitational force vector, $h(q, \dot{q})$ is a 6×1 Coriolis and centrifugal force vector, and D(q) is a 6×6 acceleration-related matrix.

Since D(q) is always nonsingular, $\ddot{q}(t)$ can be obtained from (5.6) and substituted into (5.5) to obtain the accelerations of the manipulator hand,

$$\begin{bmatrix} \dot{v}(t) \\ \dot{\Omega}(t) \end{bmatrix} = [\dot{J}(q, \dot{q})][J^{-1}(q)] \begin{bmatrix} v(t) \\ \Omega(t) \end{bmatrix} + [J(q)][D^{-1}(q)]\left[\tau(t) - h(q, \dot{q}) - c(q)\right] \tag{5.7}$$

where q and $\dot{q}$ can be obtained from Eq. (2.4b) and Eq. (2.15), respectively.

For convenience, let us partition J(q), $J^{-1}(q)$, and $D^{-1}(q)$ into 3×3 submatrices and $h(q, \dot{q})$, c(q), and $\tau(t)$ into 3×1 submatrices,

$$[J(q)] \triangleq \begin{bmatrix} J_{11}(q) & | & J_{12}(q) \\ — & | & — \\ J_{21}(q) & | & J_{22}(q) \end{bmatrix} \quad ; \quad [J^{-1}(q)] \triangleq [K(q)] \triangleq \begin{bmatrix} K_{11}(q) & | & K_{12}(q) \\ — & | & — \\ K_{21}(q) & | & K_{22}(q) \end{bmatrix} \tag{5.8}$$

$$[D^{-1}(q)] \triangleq [E(q)] \triangleq \begin{bmatrix} E_{11}(q) & | & E_{12}(q) \\ — & | & — \\ E_{21}(q) & | & E_{22}(q) \end{bmatrix} \quad ; \quad [h(q, \dot{q})] \triangleq \begin{bmatrix} h_1(q, \dot{q}) \\ — \\ h_2(q, \dot{q}) \end{bmatrix} \tag{5.9}$$

$$[c(q)] \triangleq \begin{bmatrix} c_1(q) \\ — \\ c_2(q) \end{bmatrix} \quad ; \quad [\tau(t)] \triangleq \begin{bmatrix} \tau_1(t) \\ — \\ \tau_2(t) \end{bmatrix} \tag{5.10}$$

Combining Eqs. (5.2), (5.4) and (5.7), and using Eqs. (5.8)-(5.10), we can obtain the equations of motion of the manipulator in Cartesian coordinates:

$$\begin{bmatrix} \dot{p}(t) \\ \dot{\Phi}(t) \\ \hline \dot{v}(t) \\ \Omega(t) \end{bmatrix} = \left[\begin{array}{cc|c|c} 0 & 0 & I_3 & 0 \\ 0 & 0 & 0 & M(\Phi) \\ \hline 0 & 0 & \dot{J}_{11}(q,\dot{q})K_{11}(q) + \dot{J}_{12}(q,\dot{q})K_{21}(q) & \dot{J}_{11}(q,\dot{q})K_{12}(q) + \dot{J}_{12}(q,\dot{q})K_{22}(q) \\ 0 & 0 & \dot{J}_{21}(q,\dot{q})K_{11}(q) + \dot{J}_{22}(q,\dot{q})K_{21}(q) & \dot{J}_{21}(q,\dot{q})K_{12}(q) + \dot{J}_{22}(q,\dot{q})K_{22}(q) \end{array}\right] \begin{bmatrix} p(t) \\ \Phi(t) \\ \hline v(t) \\ \Omega(t) \end{bmatrix}$$

(5.11)

$$+ \left[\begin{array}{c|c} 0 & 0 \\ 0 & 0 \\ \hline J_{11}(q)E_{11}(q) + J_{12}(q)E_{21}(q) & J_{11}(q)E_{12}(q) + J_{12}(q)E_{22}(q) \\ J_{21}(q)E_{11}(q) + J_{22}(q)E_{21}(q) & J_{21}(q)E_{12}(q) + J_{22}(q)E_{22}(q) \end{array}\right] \begin{bmatrix} -h_1(q,\dot{q}) - c_1(q) + r_1(t) \\ \hline -h_2(q,\dot{q}) - c_2(q) + r_2(t) \end{bmatrix}$$

where I_3 is a 3×3 identity matrix. The Eq. (5.11) shows that the equations of motion of the manipulator in Cartesian coordinates are extremely complex and shows the nonlinearly coupled dynamics of the manipulator. Design of an appropriate control law based on the Eq. (5.11) may be very difficult. Instead a linearized perturbation model about the nominal hand trajectory may provide a better approach.

5.4. Resolved Motion Adaptive Control

Defining the state vector for the manipulator hand as:

$$\begin{aligned} \mathbf{x}(t) &\triangleq (x_1(t), x_2(t), \cdots, x_{12}(t))^T \\ &\triangleq (p_x(t), p_y(t), p_z(t), \alpha(t), \beta(t), \gamma(t), v_x(t), v_y(t), v_z(t), \omega_x(t), \omega_y(t), \omega_z(t))^T \\ &\triangleq (\mathbf{p}^T(t), \Phi^T(t), \mathbf{v}^T(t), \Omega^T(t))^T \end{aligned} \tag{5.12}$$

and the input torque vector as:

$$\mathbf{u}(t) \triangleq (u_1(t), \cdots, u_6(t))^T \triangleq (\tau_1(t), \cdots, \tau_6(t))^T \tag{5.13}$$

the Eq. (5.11) can be expressed in state space representation as,

$$\dot{\mathbf{x}}(t) = \mathbf{f}(\mathbf{x}(t), \mathbf{u}(t)) \tag{5.14}$$

where $\mathbf{x}(t) \in R^{2n}$, $\mathbf{u}(t) \in R^{n}$, $\mathbf{f}(\cdot)$ is defined and continuously differentiable on $R^{2n} \times R^{n}$ except at certain critical points of $R^{2n} \times R^{n}$, and n is the number of degree of freedom of the manipulator. The critical points may come from the particular configuration of the manipulator and the singularity of the Euler angle formulation of the hand orientation. From Eq. (3.3), we have:

$$(\mathbf{p}^T(t), \Phi^T(t))^T = (x_1, \cdots, x_6)^T = \mathbf{N}(\mathbf{q}(t)) \tag{5.15}$$

The Eq. (5.14) can be expressed explicitly as,

$$\begin{aligned}
\dot{x}_1(t) &= f_1(\mathbf{x}(t), \mathbf{u}(t)) = x_7(t) \\
\dot{x}_2(t) &= f_2(\mathbf{x}(t), \mathbf{u}(t)) = x_8(t) \\
\dot{x}_3(t) &= f_3(\mathbf{x}(t), \mathbf{u}(t)) = x_9(t) \\
\dot{x}_4(t) &= f_4(\mathbf{x}(t), \mathbf{u}(t)) = \sec x_5(x_{10} Cx_6 + x_{11} Sx_6) \\
\dot{x}_5(t) &= f_5(\mathbf{x}(t), \mathbf{u}(t)) = -\sec x_5(x_{10} Cx_5 Sx_6 - x_{11} Cx_5 Cx_6) \\
\dot{x}_6(t) &= f_6(\mathbf{x}(t), \mathbf{u}(t)) = \sec x_5(x_{10} Sx_5 Cx_6 + x_{11} Sx_5 Sx_6 + x_{12} Cx_5) \\
\dot{x}_{i+6}(t) &= f_{i+6}(\mathbf{x}(t), \mathbf{u}(t)) = g_{i+6}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{x}(t) + b_{i+6}(\mathbf{q})\lambda(\mathbf{q}, \dot{\mathbf{q}}) + b_{i+6}(\mathbf{q})\mathbf{u}(t)
\end{aligned} \tag{5.16}$$

where $i = 1, \cdots, 6$ and $g_{i+6}(\mathbf{q}, \dot{\mathbf{q}})$ is the $(i+6)^{th}$ row of the matrix,

$$\left[\begin{array}{cc|cc}
0 & 0 & \mathbf{I}_3 & 0 \\
0 & 0 & 0 & \mathbf{M}(\Phi) \\
\hline
0 & 0 & \dot{\mathbf{J}}_{11}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{11}(\mathbf{q}) + \dot{\mathbf{J}}_{12}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{21}(\mathbf{q}) & \dot{\mathbf{J}}_{11}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{12}(\mathbf{q}) + \dot{\mathbf{J}}_{12}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{22}(\mathbf{q}) \\
0 & 0 & \dot{\mathbf{J}}_{21}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{11}(\mathbf{q}) + \dot{\mathbf{J}}_{22}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{21}(\mathbf{q}) & \dot{\mathbf{J}}_{21}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{12}(\mathbf{q}) + \dot{\mathbf{J}}_{22}(\mathbf{q}, \dot{\mathbf{q}})\mathbf{K}_{22}(\mathbf{q})
\end{array}\right]$$

$b_{i+6}(\mathbf{q})$ is the $(i+6)^{th}$ row of the matrix:

$$\begin{bmatrix} 0 & | & 0 \\ 0 & | & 0 \\ \hline & | & \\ \mathbf{J}_{11}(\mathbf{q})\mathbf{E}_{11}(\mathbf{q}) + \mathbf{J}_{12}(\mathbf{q})\mathbf{E}_{21}(\mathbf{q}) & | & \mathbf{J}_{11}(\mathbf{q})\mathbf{E}_{12}(\mathbf{q}) + \mathbf{J}_{12}(\mathbf{q})\mathbf{E}_{22}(\mathbf{q}) \\ \mathbf{J}_{21}(\mathbf{q})\mathbf{E}_{11}(\mathbf{q}) + \mathbf{J}_{22}(\mathbf{q})\mathbf{E}_{21}(\mathbf{q}) & | & \mathbf{J}_{21}(\mathbf{q})\mathbf{E}_{12}(\mathbf{q}) + \mathbf{J}_{22}(\mathbf{q})\mathbf{E}_{22}(\mathbf{q}) \end{bmatrix}$$

Moreover,

$$\lambda(\mathbf{q}, \dot{\mathbf{q}}) = \begin{bmatrix} -\mathbf{h}_1(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{c}_1(\mathbf{q}) \\ \hline -\mathbf{h}_2(\mathbf{q}, \dot{\mathbf{q}}) - \mathbf{c}_2(\mathbf{q}) \end{bmatrix},$$

$\mathbf{q} = \mathbf{N}^{-1}(x_1, \cdots, x_6)$, and from Eq. (2.15)

$$\dot{\mathbf{q}} = \mathbf{J}^{-1}(\mathbf{N}^{-1}(x_1, \cdots, x_6)) \cdot (x_7, \cdots, x_{12})^T.$$

The variables $x_1, \cdots, x_6$ must be restricted to a set on which the inverse functions $\mathbf{N}^{-1}$ and $\mathbf{J}^{-1}$ are defined and Eq. (5.16) is defined.

5.4.1. Perturbation Equations of Motion

Since the Eq. (5.16) describes the complete robot arm dynamics, the desired joint torques for each hand trajectory set point $(\mathbf{p}^d(t), \Phi^d(t), \mathbf{v}^d(t), \Omega^d(t), \dot{\mathbf{v}}^d(t), \dot{\Omega}^d(t))$ can be computed in open-loop fashion quite accurately as follows: (i) The hand trajectory set points are resolved into a set of values of desired joint positions, velocities and accelerations using Eqs. (2.4b), (2.15), (2.17), and inverse kinematics equations at each sampling instant (In fact, these joint values are obtained from the trajectory planner developed in Chapter 3.), (ii) the desired joint torques along the hand trajectory are computed from the Newton-Euler equations of motion [38] using the computed sets of values

of joint positions, velocities and accelerations. These computed torques constitute the nominal torque values.

Using the Taylor series expansion on the Eq. (5.16) about the nominal hand trajectory and assuming the higher order terms are negligible, the associated linearized perturbation model for this control system can be obtained,

$$\delta\dot{\mathbf{x}}(t) = \mathbf{A}(t)\,\delta\mathbf{x}(t) + \mathbf{B}(t)\,\delta\mathbf{u}(t) + \mathbf{e}(t) \tag{5.17}$$

where $\mathbf{A}(t) \equiv \nabla_{\mathbf{x}} \mathbf{f}\,|_{\mathbf{x}_n}$ and $\mathbf{B}(t) \equiv \nabla_{\mathbf{u}} \mathbf{f}\,|_{\mathbf{u}_n}$ are the system parameters and are equivalent to the Jacobians of $\mathbf{f}(\mathbf{x}(t), \mathbf{u}(t))$ evaluated at the nominal states, $\mathbf{x}_n(t)$, and inputs, $\mathbf{u}_n(t)$, respectively, $\mathbf{e}(t)$ is the error generated from the higher order terms of the Tayler series expansion and the inaccurate perturbation model, $\delta\mathbf{x}(t) = \mathbf{x}(t) - \mathbf{x}_n(t)$, and $\delta\mathbf{u}(t) = \mathbf{u}(t) - \mathbf{u}_n(t)$. $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are functionally defined by:

$$\mathbf{A}(t) = \left[\begin{array}{c|c} \mathbf{A}_{11}(t) & \mathbf{A}_{12}(t) \\ \hline \mathbf{A}_{21}(t) & \mathbf{A}_{22}(t) \end{array}\right] \;;\; \mathbf{B}(t) = \left[\begin{array}{c|c} 0 & 0 \\ \hline \mathbf{B}_{21}(t) & \mathbf{B}_{22}(t) \end{array}\right] \tag{5.18}$$

where the submatrices are found explicitly as,

$$\mathbf{A}_{11}(t) = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \dfrac{\partial f_4}{\partial x_5} & \dfrac{\partial f_4}{\partial x_6} \\ 0 & 0 & 0 & 0 & \dfrac{\partial f_5}{\partial x_5} & \dfrac{\partial f_5}{\partial x_6} \\ 0 & 0 & 0 & 0 & \dfrac{\partial f_6}{\partial x_5} & \dfrac{\partial f_6}{\partial x_6} \end{bmatrix} \;;\; \mathbf{A}_{12}(t) = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & \dfrac{\partial f_4}{\partial x_{10}} & \dfrac{\partial f_4}{\partial x_{11}} & 0 \\ 0 & 0 & 0 & \dfrac{\partial f_5}{\partial x_{10}} & \dfrac{\partial f_5}{\partial x_{11}} & 0 \\ 0 & 0 & 0 & \dfrac{\partial f_6}{\partial x_{10}} & \dfrac{\partial f_6}{\partial x_{11}} & \dfrac{\partial f_6}{\partial x_{12}} \end{bmatrix} \tag{5.19}$$

$$\mathbf{A}_{21}(t) = \begin{bmatrix} \frac{\partial f_7}{\partial x_1} & \frac{\partial f_7}{\partial x_2} & \cdots & \frac{\partial f_7}{\partial x_6} \\ \frac{\partial f_8}{\partial x_1} & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \frac{\partial f_{12}}{\partial x_1} & \cdot & \cdots & \frac{\partial f_{12}}{\partial x_6} \end{bmatrix} ; \mathbf{A}_{22}(t) = \begin{bmatrix} \frac{\partial f_7}{\partial x_7} & \frac{\partial f_7}{\partial x_8} & \cdots & \frac{\partial f_7}{\partial x_{12}} \\ \frac{\partial f_8}{\partial x_7} & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \cdot & \cdot & \cdots & \cdot \\ \frac{\partial f_{12}}{\partial x_7} & \cdot & \cdots & \frac{\partial f_{12}}{\partial x_{12}} \end{bmatrix} \tag{5.20}$$

$$\mathbf{B}_{21}(t) = \begin{bmatrix} \frac{\partial f_7}{\partial u_1} & \frac{\partial f_7}{\partial u_2} & \frac{\partial f_7}{\partial u_3} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \frac{\partial f_{12}}{\partial u_1} & \frac{\partial f_{12}}{\partial u_2} & \frac{\partial f_{12}}{\partial u_3} \end{bmatrix} ; \mathbf{B}_{22}(t) = \begin{bmatrix} \frac{\partial f_7}{\partial u_4} & \frac{\partial f_7}{\partial u_5} & \frac{\partial f_7}{\partial u_6} \\ \cdot & \cdot & \cdot \\ \cdot & \cdot & \cdot \\ \frac{\partial f_{12}}{\partial u_4} & \frac{\partial f_{12}}{\partial u_5} & \frac{\partial f_{12}}{\partial u_6} \end{bmatrix} \tag{5.21}$$

Relative to the closed-loop dynamics, the matrices $\mathbf{A}(t)$ and $\mathbf{B}(t)$ are slowly time-varying and depend on the instantaneous manipulator hand position and velocity. The design of a feedback control law for the perturbation equations requires that the system parameters of the Eq. (5.17) be known at all times. Thus parameter identification techniques must be used to identify the unknown elements in $\mathbf{A}(t)$ and $\mathbf{B}(t)$.

As a result of this formulation, the control problem is reduced to determining $\delta\mathbf{u}(t)$ which drives $\delta\mathbf{x}(t)$ to zero at all times along the nominal hand trajectory. In other words, the position/orientation and velocity errors of the hand along the preplanned hand trajectory are minimized. The overall controlled system is

characterized by feedforward and feedback components. The feedforward component computes the nominal joint torques $\mathbf{u}_n(t)$ from the Newton-Euler equations of motion and the feedback component computes the perturbation torques $\delta\mathbf{u}(t)$. The main advantages of this formulation are twofold. Firstly it reduces a nonlinear control problem to a linear control problem about a nominal hand trajectory, and secondly the computations of the nominal and perturbation torques can be performed separately and simultaneously.

5.4.2. Parameter Identification and Perturbation Control

Hereafter, we denote n as the degree of freedom of the robot. For implementation on digital computer, the Eq. (5.17) needs to be discretized to obtain the following discrete-time linear equations appropriate for parameter identification:

$$\delta\mathbf{x}(\,(k+1)T\,)=\mathbf{F}(\,kT\,)\,\delta\mathbf{x}(\,kT\,)+\mathbf{G}(\,kT\,)\,\delta\mathbf{u}(\,kT\,)+\mathbf{E}(\,kT\,)\quad;\quad k=0,1,\cdots \tag{5.22}$$

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

$$\delta\mathbf{x}(\,kT\,)=\Theta(\,kT\,,t_0\,)\,\delta\mathbf{x}(t_0)+\int_{t_0}^{kT}\Theta(\,kT\,,t\,)(\,\mathbf{B}(\,t\,)\,\delta\mathbf{u}(\,t\,)+\mathbf{e}(\,t\,)\,)\,dt \tag{5.23}$$

where $\Theta(\,kT\,,t_0\,)$ is the state-transition matrix of the system (5.17). $\mathbf{F}(\,kT\,)$, $\mathbf{G}(\,kT\,)$, and $\mathbf{E}(\,kT\,)$ are respectively $2n\times 2n$, $2n\times n$, and $2n\times 1$ matrices and are given by:

$$\mathbf{F}(\,kT\,)=\Theta(\,(k+1)T\,,kT\,) \tag{5.24}$$

$$\mathbf{G}(kT) = \int_{kT}^{(k+1)T} \Theta((k+1)T, t)\,\mathbf{B}(t)\,dt \tag{5.25}$$

and

$$\mathbf{E}(kT) = \int_{kT}^{(k+1)T} \mathbf{e}(t)\,dt \tag{5.26}$$

With this model, a maximum of $6n^2$ parameters need to be identified. Without confusion, the sampling period T will be dropped from the above equations for clarity. Specifically, we abuse notations and let $\delta\mathbf{x}(kT) = \mathbf{x}(k)$, $\delta\mathbf{u}(kT) = \mathbf{u}(k)$, and $\mathbf{E}(kT) = \mathbf{E}(k)$. Then we obtain the following equation from Eq. (5.22).

$$\mathbf{x}(k+1) = \mathbf{F}(k)\,\mathbf{x}(k) + \mathbf{G}(k)\,\mathbf{u}(k) + \mathbf{E}(k) \quad ; \quad k = 0, 1, \cdots \tag{5.27}$$

In order to simplify the identification algorithm for real-time applications, a recursive least square parameter identification scheme is used. In the parameter identification scheme, we make the following assumptions: 1) The parameters of the system are slowly time-varying but the variation speed is slower than the adaptation speed, 2) Measurement noise is negligible, and 3) The state variables $\mathbf{x}(k)$ of Eq. (5.27) are measurable.

The first assumption was justified from the simulation in the joint variable space control [25] and the same findings were reported in a recent publication [23]. The second assumption was made to simplify the identification scheme and it needs to be verified experimentally. The third assumption was based on the fact that the state variable $\mathbf{x}(k)$ can be computed from the measured values of $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ and the desired hand position and velocity from the hand trajectory.

Defining and putting the i^{th} row of the unknown system parameters in the Eq. (5.27) at the k^{th} instant of time in a $3n+1$–dimensional vector as,

$$\theta_i(k) = (f_{i1}(k), \cdots, f_{ip}(k), g_{i1}(k), \cdots, g_{in}(k), e_i(k))^T \tag{5.28}$$

where $p = 2n$, and the outputs and inputs at the k^{th} instant of time in a vector as,

$$\phi(k) = (x_1(k), \cdots, x_p(k), u_1(k), \cdots, u_n(k), 1)^T \tag{5.29}$$

and the state vector at the k^{th} instant of time as,

$$\mathbf{x}(k) = (x_1(k), \cdots, x_p(k))^T \tag{5.30}$$

Then, the Eq. (5.27) can be rewritten as follow,

$$x_i(k+1) = \phi^T(k)\theta_i(k) \qquad ; \; i = 1, 2, \cdots 2n. \tag{5.31}$$

where $x_i(k+1)$ is the i^{th} state variable at the $(k+1)^{th}$ sampling instant.

Based on the input-output relation in Eq. (5.31), a recursive least squares parameter identification algorithm can be found [14,52]:

$$\begin{aligned} \hat{\theta}_i(k+1) &= \hat{\theta}_i(k) + \mathbf{P}(k)\phi(k) \, [\phi^T(k)\mathbf{P}(k)\phi(k) + \rho]^{-1} \; [x_i(k+1) - \phi^T(k)\hat{\theta}_i(k)] \\ \mathbf{P}(k+1) &= [\mathbf{P}(k) - \mathbf{P}(k)\phi(k) \, [\phi^T(k)\mathbf{P}(k)\phi(k) + \rho]^{-1} \, \phi^T(k)\mathbf{P}(k)] \end{aligned} \tag{5.32}$$

for $i = 1, 2, \cdots, 2n$, where $0 < \rho < 1$ and "hat" is used to indicate the estimate of the parameters $\theta_i(k)$ and $\mathbf{P}(k)$ is a $(3n+1)\times(3n+1)$ symmetric positive definite matrix. $\mathbf{P}(k) = \rho \, [\Psi(k) \, \Psi^T(k)]^{-1}$ and $\Psi(k) = [\phi(1), \phi(2), \cdots, \phi(k)]$ is the measurement matrix up to the k^{th} sampling instant. The $\mathbf{P}(k)$ matrix has the similar effect as the error covariance matrix in stochastic identification with zero mean modeling error.

The above recursive equations indicate that the estimate of the parameters $\hat{\theta}_i(k+1)$ at the $k+1^{th}$ sampling period is equal to the previous estimate $\hat{\theta}_i(k)$ corrected by the term proportional to $[x_i(k+1) - \phi^T(k)\hat{\theta}_i(k)]$. The $\phi^T(k)\hat{\theta}_i(k)$ is

the prediction of the value $x_i(k+1)$ based on the estimate of the parameters $\theta_i(k)$ and the measurement vector $\phi(k)$. The components of the vector $P(k)\phi(k)[\phi^T(k)P(k)\phi(k) + \rho]^{-1}$ are weighting factors which indicate how the corrections and the previous estimate should be weighted to obtain the new estimate $\hat{\theta}_i(k+1)$.

The parameter ρ is a weighting factor which is commonly used for tracking slowly time-varying parameters by exponentially forgetting the "aged" measurements. If $\rho \ll 1$, a large weighting factor is placed on the more recent sampled data by rapidly weighing out previous samples. If $\rho \approx 1$, accuracy in tracking the the time-varying parameters will be lost due to the truncation of measured data sequence. We can compromise between fast adaptation capabilities and loss of accuracy in parameter identification by adjusting the weighting factor ρ. In most applications for tracking slowly time-varying parameters, ρ is usually chosen to be $0.90 \leq \rho < 1.0$.

Finally the above identification scheme can be started by choosing the initial values of P(0) to be

$$P(0) = \sigma I_{3n+1} \tag{5.33}$$

where σ is a large positive scalar. The initial estimate of the unknown parameters $F(k)$ and $G(k)$ can be approximated by the following equations:

$$F(0) \approx I_{2n} + \left[\frac{\partial f}{\partial x}(x_n(0),u_n(0))\right]T + \left[\frac{\partial f}{\partial x}(x_n(0),u_n(0))\right]^2 \frac{T^2}{2} \tag{5.34a}$$

$$\mathbf{G}(0) \approx \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right] T + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]\left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right] T^2 + \left[\frac{\partial \mathbf{f}}{\partial \mathbf{x}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right]^2 \left[\frac{\partial \mathbf{f}}{\partial \mathbf{u}}(\mathbf{x}_n(0),\mathbf{u}_n(0))\right] \frac{T^3}{2} \tag{5.34b}$$

where T is the sampling period.

With the determination of $\mathbf{F}(k)$ and $\mathbf{G}(k)$ from the parameter identification scheme, proper control laws can be designed to obtain the required correction torques to reduce the position/orientation and velocity errors of the manipulator hand along a nominal hand trajectory. This can be done by finding an optimal control, $\mathbf{u}^*(k)$, which minimizes the performance index, $J(k)$, while satisfying the constraints of the Eq. (5.27):

$$J(k) = \frac{1}{2}\left[\mathbf{x}^T(k+1)\mathbf{Q}\mathbf{x}(k+1) + \mathbf{u}^T(k)\mathbf{R}\mathbf{u}(k)\right] \tag{5.35}$$

where $\mathbf{Q}$ is a $2n \times 2n$ semi-positive definite weighting matrix and $\mathbf{R}$ is an $n \times n$ positive definite weighting matrix. We can use the diagonal matrix for $\mathbf{Q}$ which can be compromised between the tracking accuracy and the allowable magnitude of perturbed inputs. Also, we can constrain the perturbed torque by increasing the elements of the weighting matrix $\mathbf{R}$. The one-stage performance index in the Eq. (5.35) indicates that the objective of the optimal control is to drive the position/orientation and velocity errors of the manipulator hand to zero along the nominal hand trajectory in a coordinated position and rate control per interval step, while at the same time, a cost is attached to the use of control efforts. The optimal control solution which minimizes the functional in the Eq. (5.35) subject to the constraints of the Eq. (5.27) is well-known and is found to be [52]:

$$\mathbf{u}'(k) = -\left[\mathbf{R} + \hat{\mathbf{G}}^T(k)\,\mathbf{Q}\,\hat{\mathbf{G}}(k)\right]^{-1}\hat{\mathbf{G}}^T(k)\,\mathbf{Q}\,\hat{\mathbf{F}}(k)\,\mathbf{x}(k) \qquad (5.36)$$

where $\hat{\mathbf{F}}(k)$ and $\hat{\mathbf{G}}(k)$ are the system parameters obtained from the identification algorithm at the k^{th} sampling instant.

From the simulation of the adaptive perturbation control in the joint variable space [25], given the $\mathbf{F}(0)$ and $\mathbf{G}(0)$ matrices in the Eqs. (5.34a) and (5.34b), it was found that the performance of the adaptive controller is quite sensitive to the initial values in the $\mathbf{P}(0)$ matrix and less sensitive to the $\mathbf{Q}$ and $\mathbf{R}$ weighting matrices. This result is expected as the $\mathbf{P}(k)$ matrix has the same effect as the error covariance matrix in stochastic identification with zero mean modeling error, if the weighting factor ρ is chosen appropriately.

In general, the convergence of the parameter identification is not guaranteed and hence the performance of the adaptive controller depends on the proper selection of the $\mathbf{P}(0)$ matrix. The initial large values of the $\mathbf{P}(0)$ matrix usually give better convergence. The values of the $\mathbf{Q}$ matrix did not affect the performance of the controller very much, however, a small value of the $\mathbf{R}$ matrix did give a better tracking result. These results agree with the findings in [23].

The above identification and control algorithms in the Eqs. (5.32) and (5.36) do not require complex computations. In the Eq. (5.32), $[\phi^T(k)\mathbf{P}(k)\phi(k) + \rho]$ gives a scalar value which simplifies its inversion. Although the weighting factor ρ can be adjusted for each i^{th} state variable as desired, this requires excessive computations in the $\mathbf{P}(k+1)$ matrix. For real-time robot arm control, such adjustment is not desirable. $\mathbf{P}(k+1)$ is computed only once at each sampling time using the same weighting factor ρ. Moreover, since $\mathbf{P}(k)$ is a symmetric positive definite matrix, only the upper diagonal matrix of $\mathbf{P}(k)$ needs to be computed.

The combined identification and control algorithm can be computed in $O(n^3)$ time as shown in the next section, where the computational requirements of the identification and control algorithms for a n-joint manipulator is tabulated in Table 5.1. The proposed resolved motion adaptive control block diagram is shown in Figure 5.2.

It is notable that the feedback state $\mathbf{x}(t)$ is calculated from the measurements of $\mathbf{q}(t)$ and $\dot{\mathbf{q}}(t)$ in the control block diagram. The kinematics routine and J(q) are stored in the computer, thus the control method can be viewed as a control method in the joint variable-space. An alternative way of doing control in the joint-variable space has been discussed in [25]. It eliminates the computational complexity of the kinematics routine and J(q). The disadvantage of doing this is that the error is measured in the joint-variable space and small error in the joint variables may give fairly large error in Cartesian space. By suitable choice of Q and R, it may be possible to correct this disadvantage. However, the choice of the weighting matrices will require more investigation.

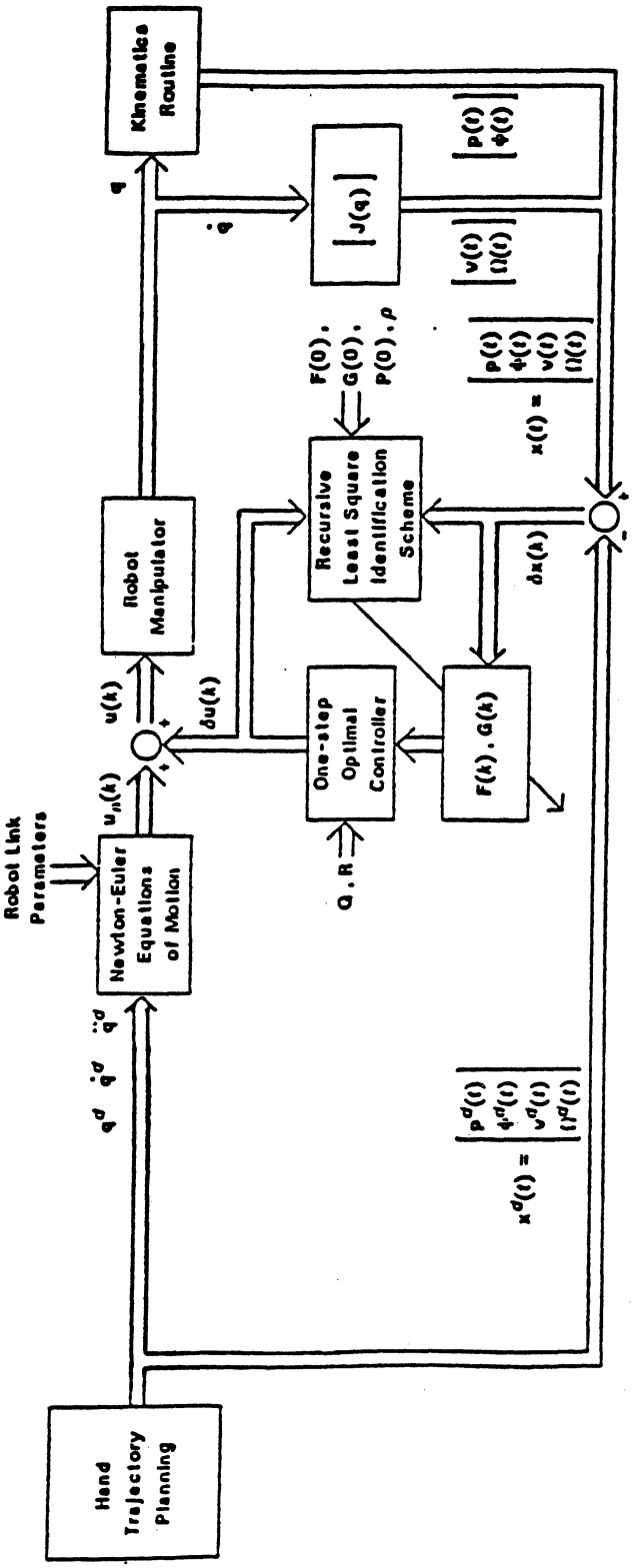


Figure 5.2 Resolved Motion Adaptive Control Block Diagram

5.5. Computational Complexity of Resolved Motion Adaptive Control

The overall resolved motion adaptive control system is characterized by feedforward and feedback components which can be computed separately and simultaneously. Assuming that multi-processors are available for parallel computations, a feasibility study of implementing the adaptive controller using present day low-cost microprocessors can be conducted by looking at the computational requirements in terms of required mathematical multiplication and addition operations.

The feedforward component which computes the nominal joint torques along a desired hand trajectory can be computed directly from the trajectory planner information because the Cartesian trajectory planner in Chapter 3 provides the trajectory information both in the joint and the Cartesian space simultaneously. The nominal joint torques can be computed along the desired hand trajectory using the Newton-Euler equations of motion [24]. It requires a total of 678 multiplications and 597 additions for a PUMA manipulator.

The feedback control component which computes the perturbation torques can be conveniently computed serially in three separate stages. The first stage computes the actual manipulator hand position and velocity from the measured values of joint position and velocity. The second stage involves the computations of the least square identification scheme (Eq. (5.32)) for the linearized perturbation system. The third stage involves the computation of the optimal adaptive self-tuning controller (Eq. (5.36)) for the linearized system. It requires about 3348 multiplications and 3118 additions for the PUMA manipulator.

Since the feedforward and feedback components can be computed in parallel, the proposed adaptive controller requires a total of 3348 multiplications and 3118

additions in each sampling period for the PUMA manipulator. Computational requirements in terms of multiplication and addition operations for the feedforward and feedback components for a n-joint manipulator are tabulated in Table 5.1.

Based on the specification sheet of an INTEL 8087, a Motorola MC68000 microprocessor, and a PDP 11/45 computer, and assuming that for each integer multiply and addition operation, it requires two memory fetches, the required time to compute the proposed adaptive controller is tabulated in Table 5.2. Both INTEL and Motorola microprocessors are not fast enough for closing the servo loop. The PDP 11/45 can compute the controller in about 17.8 *msec* which translates to approximately a sampling frequency of 56 Hz. However, the PDP 11/45 is a uniprocessor machine and parallel computation assumption is not valid. But it does give us an indication of the required processing speed of the microprocessors.

In the above computations, we neglect the time to compute the trigonometric function values, the time required to sample the joint position and velocity and the time required to perform rescaling of the joint values. However, we anticipate that faster microprocessors will be just around the corner that will be able to compute the proposed resolved motion adaptive controller in 10 *msec* within a year or two.

Adaptive Controller		Number of Multiplications	Number of Additions
stage 1	Compute τ	$117n - 24$ (678)	$103n - 21$ (597)
Total Feedforward Computations		$117n - 24$ (678)	$103n - 21$ (597)
stage 1	Compute $(p^T \cdot \Phi^T)^T$ Compute $(v^T \cdot \Omega^T)^T$	(48) $n^2 + 27n - 21$ (177)	(22) $n^2 + 18n - 15$ (129)
stage 2	Compute Hand Errors $(x(k) - x_n(k))$ and Identification Scheme	0 (0) + $33n^2 + 9n + 2$ (1244)	$2n$ (12) + $34\ ½n^2 - 1\ ½n$ (1233)
stage 3	Compute Adaptive Controller	$8n^3 + 4n^2 + n + 1$ (1879)	$8n^3 - n$ (1722)
Total Feedback Computations		$8n^3 + 38n^2 + 37n + 30$ (3348)	$8n^3 + 35\ ½n^2 + 17\ ½n + 7$ (3118)
Total Mathematical Operations		$8n^3 + 38n^2 + 37n + 30$ (3348)	$8n^3 + 35\ ½n^2 + 17\ ½n + 7$ (3118)

(Number inside the parenthesis indicates computations for n=6)

Table 5.1 Parallel Computations of the Proposed Adaptive Controller

Computing System	Integer Multiplications	Integer Additions	Memory Fetch	Required Computing Time
INTEL 8087 (5 MHz Clock)	19 μs	17 μs	9 μs	233.01 *ms* (4.29 *Hz*)
MC68000 (12.5 MHz Clock)	5.6 μs	0.96 μs	0.32 μs	25.88 *ms* (38.64 *Hz*)
PDP-11/45	3.3 μs	300 *ns*	450 *ns*	17.80 *ms* (56.17 *Hz*)

Table 5.2 Implementation of Resolved Motion Adaptive Control

5.6. Summary

A resolved motion adaptive control based on the perturbation theory has been presented. The adaptive control system is characterized by feedforward and feedback components. The feedforward component computes the nominal joint torques $\mathbf{u}_n(t)$ from the Newton-Euler equations of motion using the joint information from the trajectory planner, and the feedback component consisting of recursive least square identification and control algorithms for the linearized system computes the perturbation torques $\delta\mathbf{u}(t)$.

The analysis presented is based on the ideal system study because it neglected such nonlinear effects as gear friction and backlash. The physical implementation of the proposed adaptive control may require further investigation on the effects of gear friction, backlash, control device dynamics, and flexible link structure to the controller. Also, since the computations of the nominal and perturbation torques can be performed in parallel, the computations of the resolved motion adaptive control for a six-joint robot arm may be implemented in low-cost microprocessors. Tables 5.1 and 5.2 show the feasibility study of implementing the resolved motion adaptive control scheme.

CHAPTER 6

COMPUTER SIMULATION

6.1. Straight Line Trajectory Planning

6.1.1. Simulation and Results

A "C" language program was written on a VAX-11/780 computer to evaluate the performance of the trajectory planning scheme discussed in Chapter 3.

The trajectory planner reads in the given initial and final locations of a straight line path with the stopping variables δ_1 and δ_2, and outputs the control set points in joint variable coordinates. The control set points consist of joint position, velocity and acceleration along the given straight line path. The corresponding Cartesian control set points can be obtained also directly from the trajectory planner. The servo control period in this simulation was assumed to be 10 *msec* .

Numerical values which are tabulated in Table 6.1 were inputed into the trajectory planner. The torque constraints which depend upon the instantaneous joint position and velocity were not available from the manufacturer's specification sheet for the PUMA robot. Thus we implemented only the smoothness constraint into the simulation package. The jerk, acceleration and velocity bounds are tabulated in Table 6.2. The acceleration and velocity bounds are from the reference [34] and the jerk bound are the somewhat arbitrary set of values shown in Table 6.2.

Recall that the ρ_1 and ρ_2 must be chosen for $\Delta\lambda^{low}(k)$ and $\Delta\lambda^{high}(k)$ such that the former satisfies the constraint limits in step FW2 while the latter does not. In fact, the choice of ρ_1 and ρ_2 is closely related to the existence of straight line trajectory for a given path as discussed in Section 3.4.2. The ρ_1 and ρ_2 were selected to be 1 and 2, respectively, as a choice in the algorithm FW. Also, the ρ_1 and ρ_2 were selected to be 0 and 1, respectively, as a choice in the algorithm MFW.

The number of forward and backward search points was found to be 121. From the procedure BR, the best break point, which yields the smallest position and Euler angle errors among all possible break points, occurred at the 84^{th} control set point. Depending on the choice of the break point, the Cartesian and angular position, velocity and acceleration errors between the final search point and the final desired point vary appreciately. Table 6.3 indicates the final position and Euler angle errors from several possible break points for the given straight line path. The linear velocity and acceleration of the robot are also shown with the required number of relaxation points when the discrete time relaxation process is applied. It clearly indicates that there is a trade-off between the final position and Euler angle errors and the total traveling time for the given path.

From the simulation, the acceleration difference between the final search point $[\, p(k_g),\, \Phi(k_g)\,]$ and the final desired point $[\, p(k_{ft}),\, \Phi(k_{ft})\,]$ for the possible break points were found to exceed the jerk bounds of the joint motors. Thus, relaxation points are inserted between these two points. The resultant Cartesian position error due to the relaxation process was found to be less than 1 *mm* while the maximum Euler angle error was found to be less than 0.1 degree.

Table 6.4 shows more details about the effects of various break points on the final search point. Depending on the location of a break point, the Cartesian position, Euler angles, joint velocity, and joint acceleration values at the final search point are shown. From Table 6.4, if the trajectory planner starts the deceleration portion from the 84^{th} point of the trajectory set points, it produces the smallest position and Euler angle errors at the final search point among the all possible break points. Hence, the 84^{th} point from the procedure BR is the best break point for this trajectory planning.

In the output data of the simulation, all the control set points are within the smoothness constraints. Figures 6.1 - 6.6 show the joint position, velocity, and acceleration for the given straight line trajectory with relaxation points included, where markers are used for every 50 *msec*. For the figures of joint trajectories from the simulation, the integer parts of the time and joint angles are spaced from their fractional parts on the axis label. Most of joint trajectories are found to be constrained by the limits from the jerk bound and the given straight line requirement. Figure 6.4 shows that the acceleration of joint 4 reaches the maximum allowable limit for the acceleration and deceleration portion of the straight line trajectory. Although the resultant joint acceleration trajectory shows rapid changes near the break point, the acceleration set points are still within the allowable jerk bound for the smoothness of the joint trajectory.

Using Eqs. (3.1) and (3.60), the traveled length at each servo time interval can be computed. Then the traveled length versus servo time curve can be plotted for this straight line trajectory as shown in Figure 6.7. This curve will be used in the collision map to determine the collision-free path and/or trajectory in Section 6.4.

Initial Cartesian Point (m)	(-0.140000, 0.560000, 0.390000)
Initial Orientation (deg.)	(0.00000, 90.00000, 90.00000)
Initial Joint Vel. & Acc. (deg/ sec and deg/ sec^2)	$\dot{q}_i = \ddot{q}_i = 0$, i = 1,2, · · · ,n
Final Cartesian Point (m)	(0.000000, 0.440000, 0.480000)
Final Orientation (deg.)	(30.00000, 60.00000, 60.00000)
Final Joint Vel. & Acc. (deg/ sec and deg/ sec^2)	$\dot{q}_i = \ddot{q}_i = 0$, i = 1,2, · · · ,n
Sampling Period	10 msec
Stopping Variables	$\delta_1 = 0.000001$, $\delta_2 = 0.0001$
Number of Forward Search Points	121
Number of Backward Search Points	121
Possible Break Point	at the 85th sampling point
Actual Break Point	at the 84th sampling point

Table 6.1 Data Used in Computer Simulation

	Joint 1	Joint 2	Joint 3	Joint 4	Joint 5	Joint 6
ϵ_i^j (deg/sec^3)	700	500	2100	4000	2100	8100
ϵ_i^a (deg/sec^2)	45	40	75	70	90	80
ϵ_i^v (deg/sec)	100	95	100	150	130	110

Table 6.2 Physical Constraints for Computer Simulation

Break Points	Final Point	Position Errors (Before Relaxation)		Vel. and Acc. Errors (Before Relaxation)		No. of Set Points	No. of Relaxation Points	Total No. of Set Points	Error (After Relaxation)	
		Position (mm)	Orientation (deg)	Velocity (mm/sec)	Acceleration (mm/sec^2)				Position (mm)	Orientation (deg)
1 to 83	NOT-REACHABLE ↑									
84	REACHABLE	0.041	0.006	11.2	303	167	4	171	0.274	0.040
85	REACHABLE	0.055	0.008	57.3	308	154	3	157	1.190	0.170
86	REACHABLE	0.789	0.115	85.6	302	147	3	150	1.581	0.321
87	REACHABLE	0.399	0.058	100.5	303	144	3	147	2.207	0.378
88 to 121	↓ REACHABLE			↓ INCREASING		↓ DECREASING		↓ DECREASING	↓ INCREASING	↓ INCREASING

Table 6.3 Comparisons of Several Break Points

Choice of Break Point	Final Search Point	p_x or Joint 1	p_y or Joint 2	p_z or Joint 3	α or Joint 4	β or Joint 5	γ or Joint 6
84	$p(k_g), \Phi(k_g)$	-0.000028	0.440024	0.479982	29.994048	60.005951	60.005951
	$\dot{q}(k_g)$	-1.043653	-0.596468	0.054300	-2.537757	-0.638623	1.227129
	$\ddot{q}(k_g)$	28.618493	16.452218	-1.5068543	69.975288	17.621888	-33.716521
85	$p(k_g), \Phi(k_g)$	-0.000037	0.440032	0.479976	29.992112	60.007891	60.007891
	$\dot{q}(k_g)$	-5.315253	-3.039434	0.274248	-12.943550	-3.258000	6.250477
	$\ddot{q}(k_g)$	27.389059	15.860980	-0.320165	69.906986	18.373209	-31.529861
86	$p(k_g), \Phi(k_g)$	-0.000538	0.440462	0.479654	29.884615	60.115384	60.115384
	$\dot{q}(k_g)$	-7.908676	-4.527156	0.385591	-19.327388	-4.878804	9.285301
	$\ddot{q}(k_g)$	25.749813	15.382867	1.094965	69.906986	18.919624	-28.592879
87	$p(k_g), \Phi(k_g)$	-0.000272	0.440233	0.479825	29.941750	60.058250	60.058250
	$\dot{q}(k_g)$	-9.297253	-5.319864	0.462254	22.690571	-5.722332	10.921216
	$\ddot{q}(k_g)$	24.725285	14.960249	1.961011	70.009439	19.397737	-26.859719
88	$p(k_g), \Phi(k_g)$	-0.000731	0.440627	0.479530	29.843251	60.156748	60.156748
	$\dot{q}(k_g)$	-11.066272	-6.336757	0.522947	-27.094677	-6.849314	12.981799
	$\ddot{q}(k_g)$	23.086039	14.471463	3.467388	69.872835	19.739247	-23.751983

Units: p is in meters, Φ is in degrees, $\dot{q}$ is in *deg/sec*, and $\ddot{q}$ is in *deg/sec*^{2}.

Table 6.4 Effects of Various Break Points on the Final Search Point

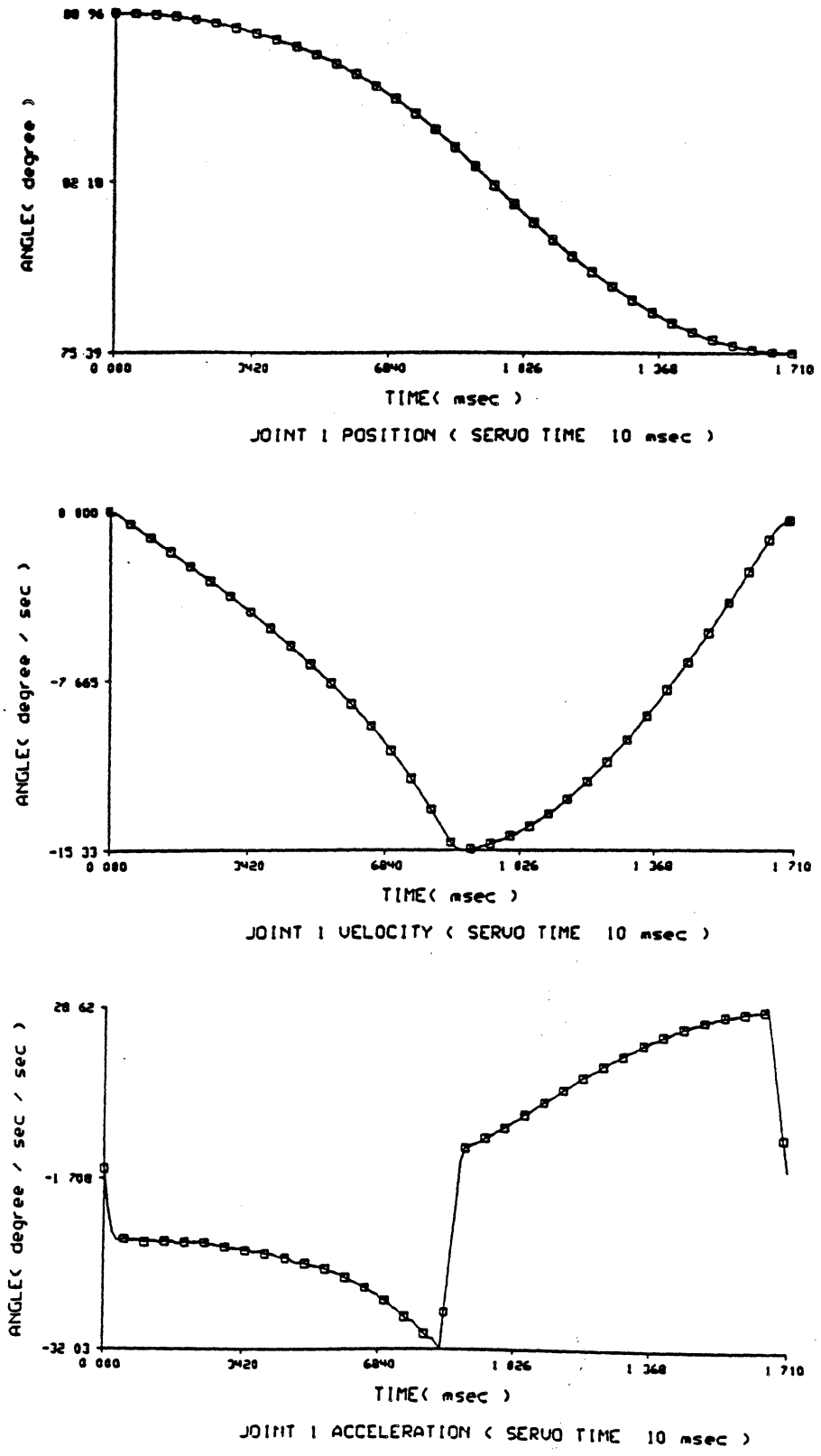


Figure 6.1 Joint 1 Trajectory of the Straight Line

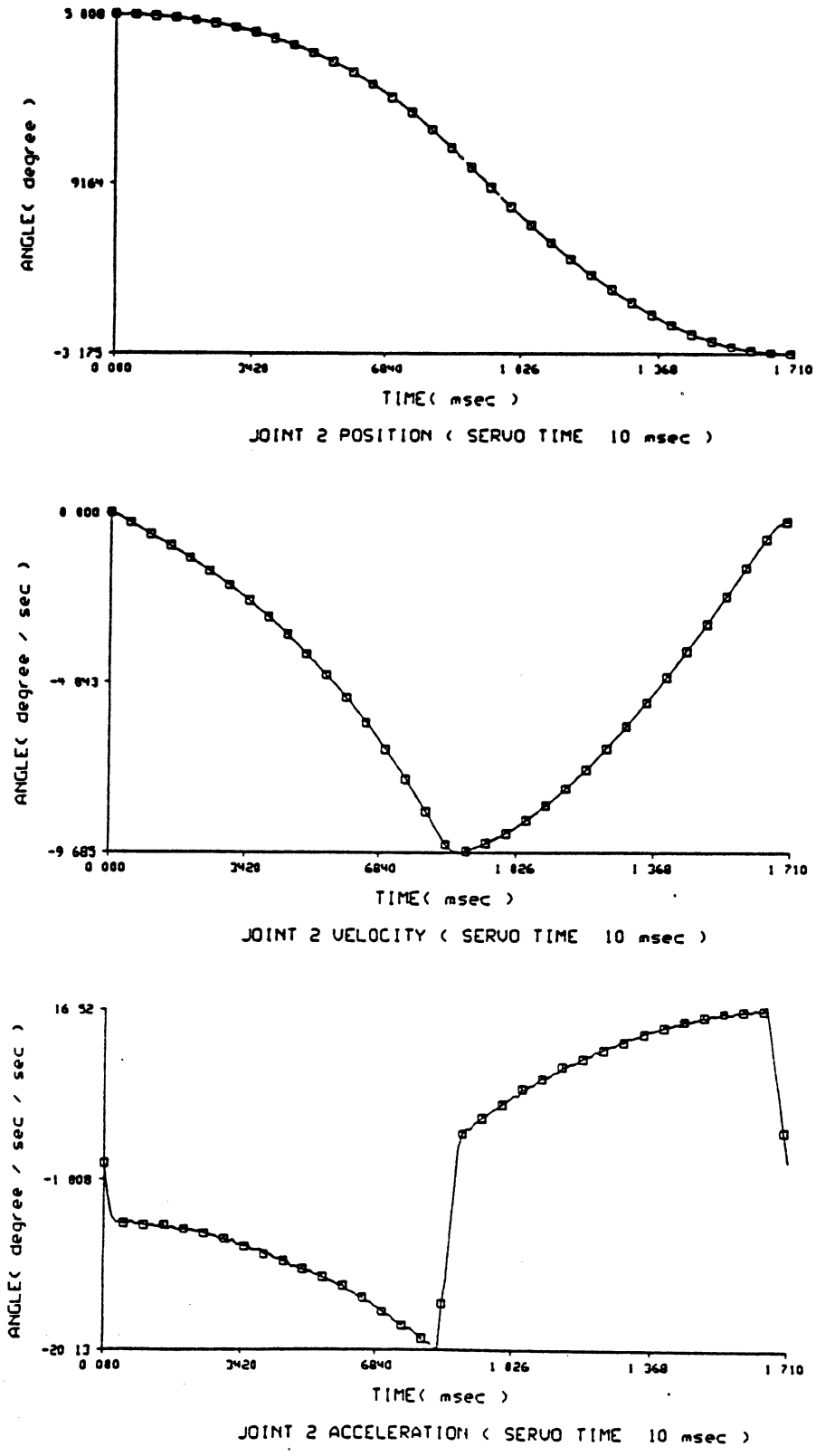


Figure 6.2 Joint 2 Trajectory of the Straight Line

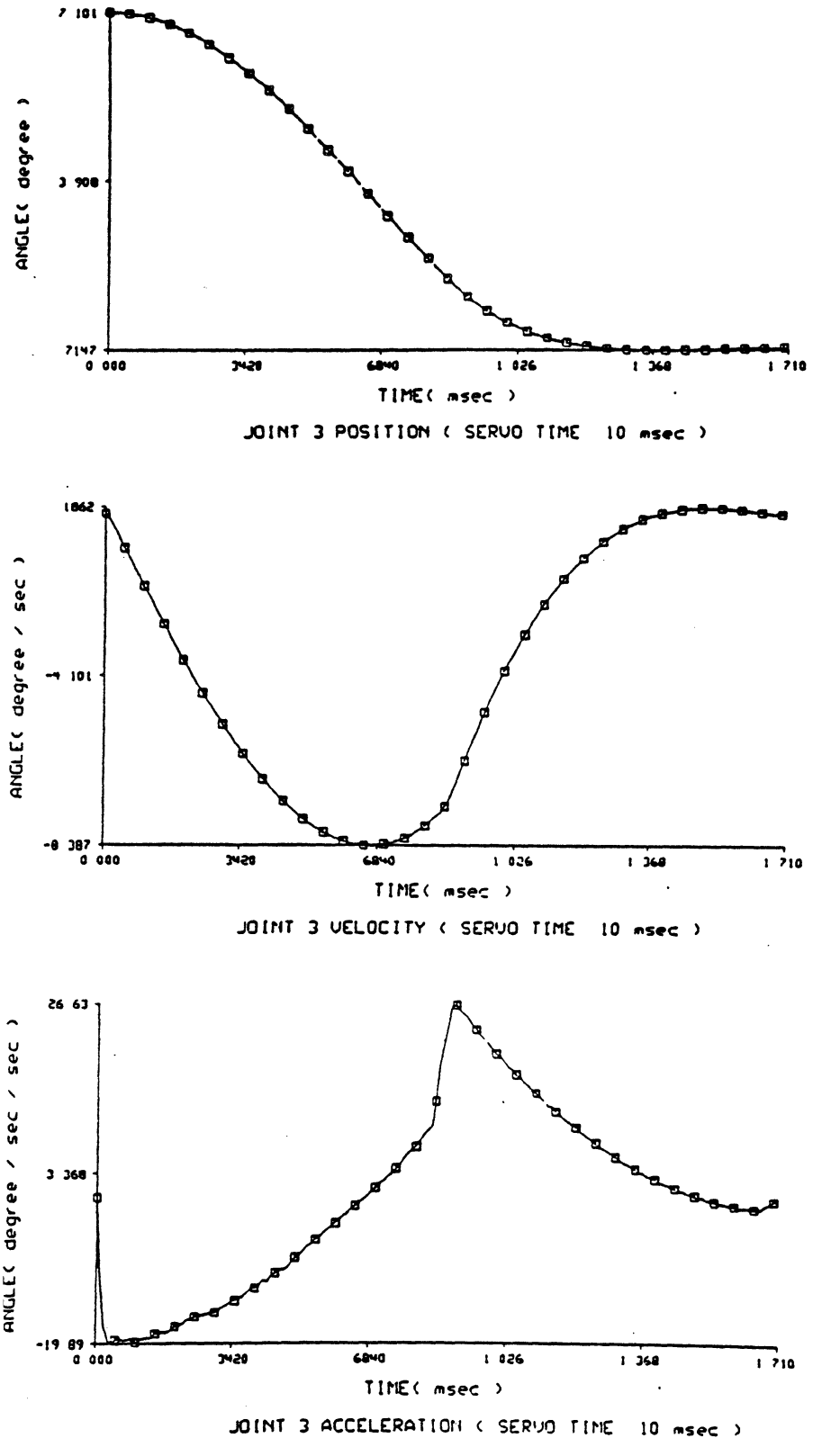


Figure 6.3 Joint 3 Trajectory of the Straight Line

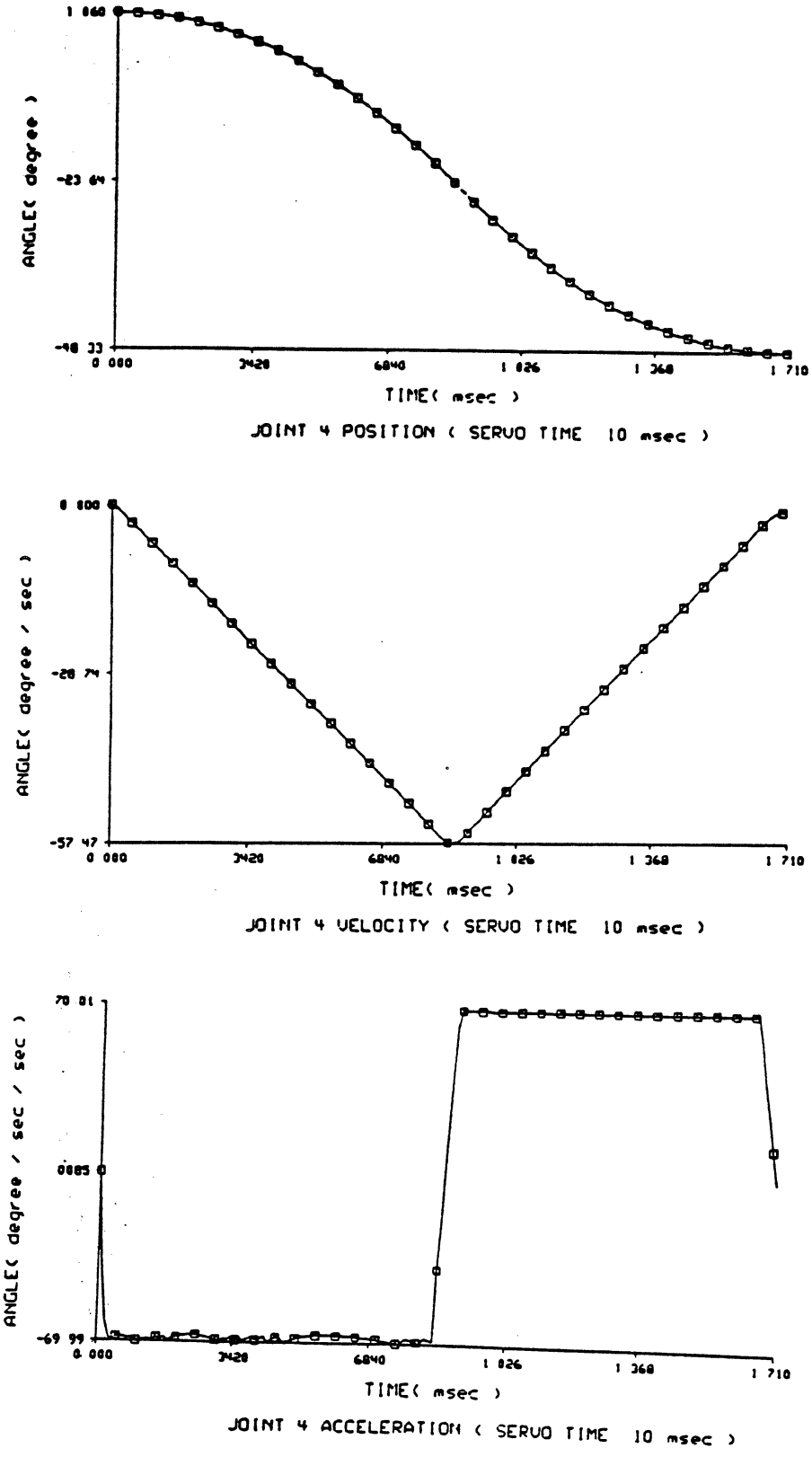


Figure 6.4 Joint 4 Trajectory of the Straight Line

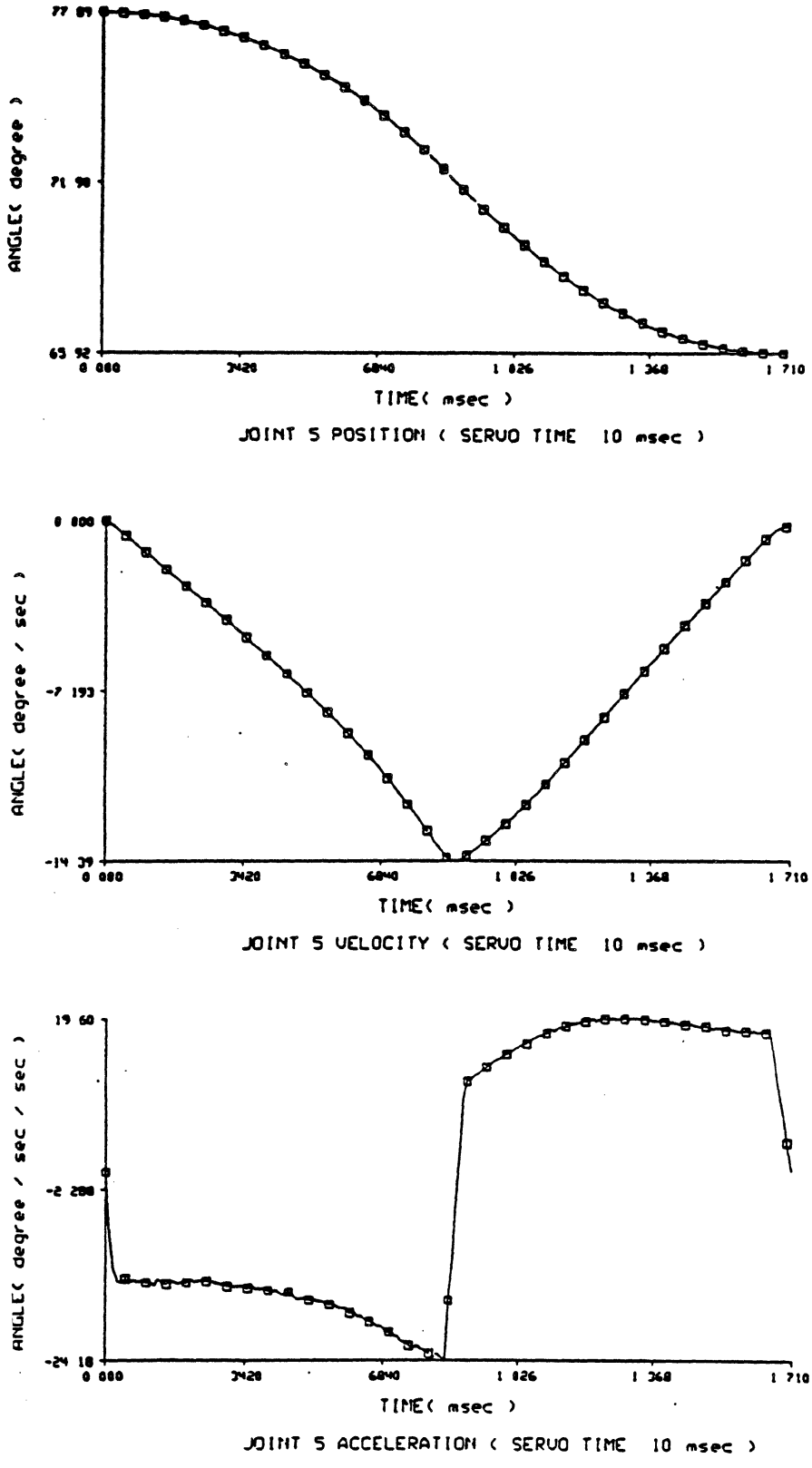
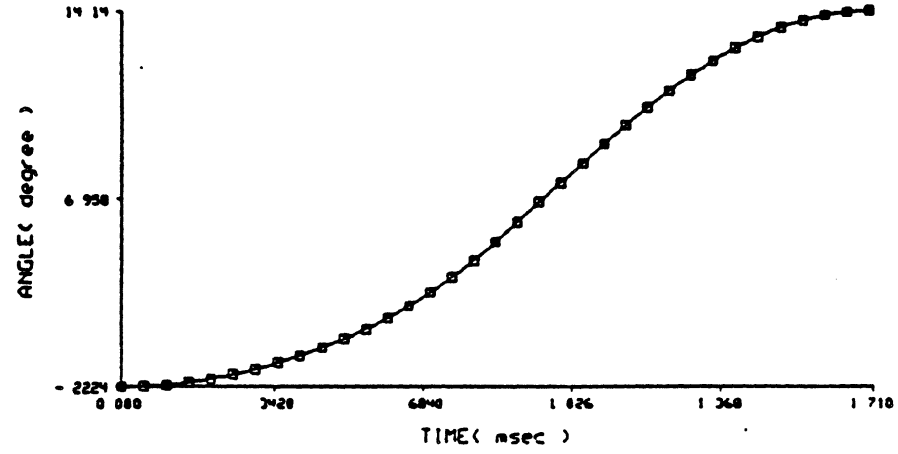
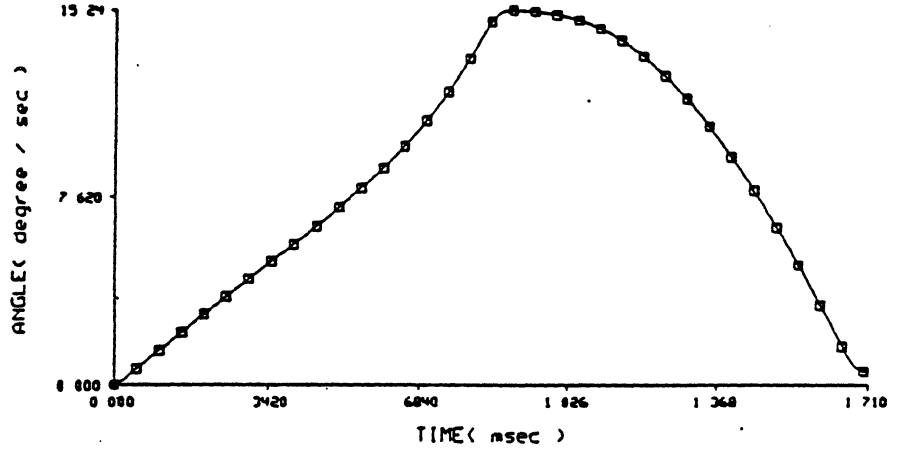


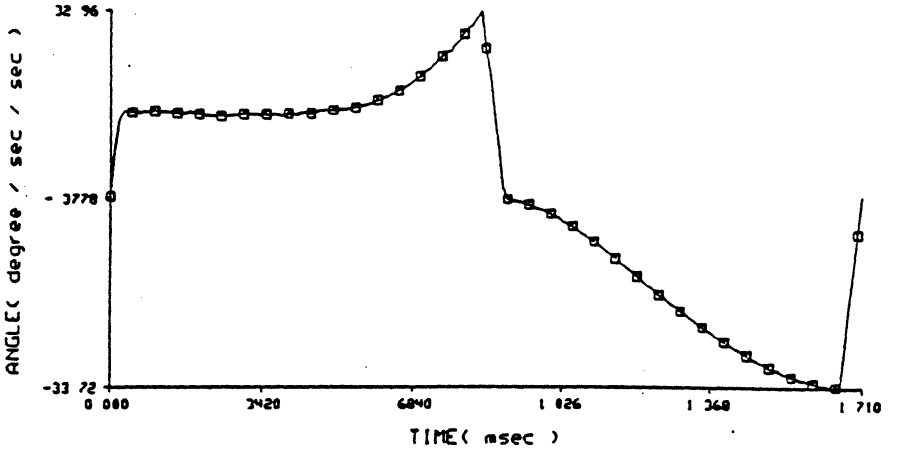
Figure 6.5 Joint 5 Trajectory of the Straight Line



JOINT 6 POSITION (SERVO TIME 10 msec)



JOINT 6 VELOCITY (SERVO TIME 10 msec)



JOINT 6 ACCELERATION (SERVO TIME 10 msec)

Figure 6.6 Joint 6 Trajectory of the Straight Line

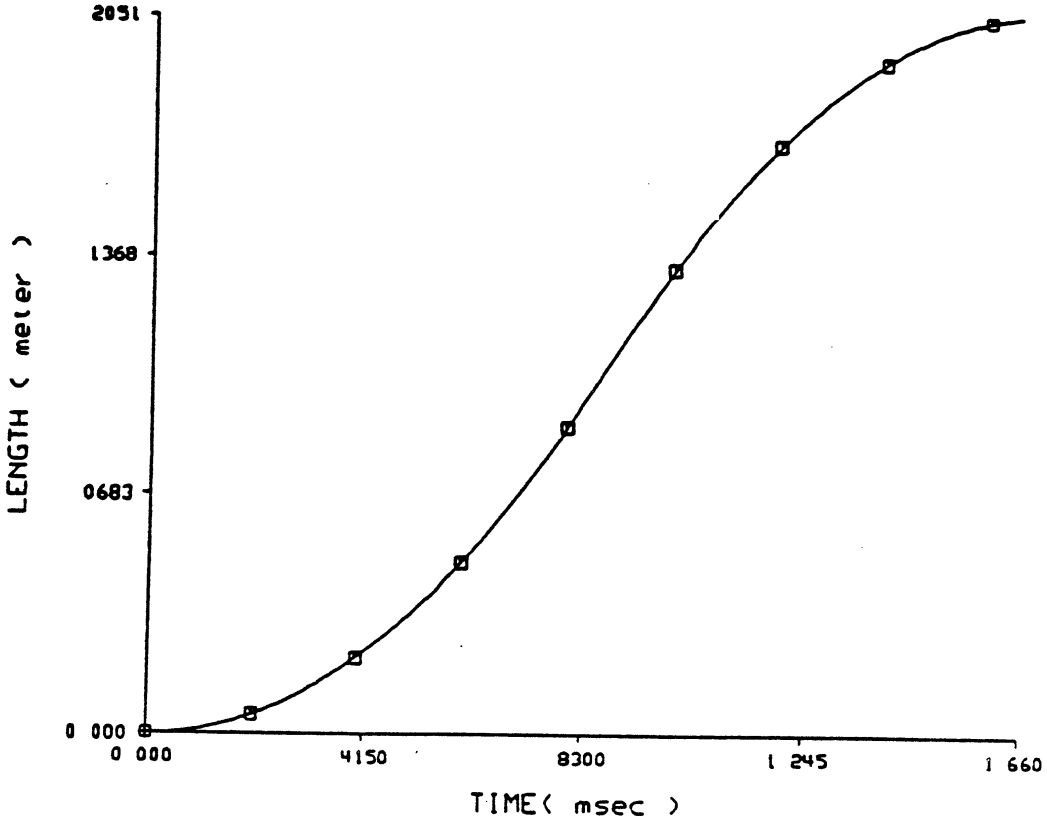


Figure 6.7 Traveled Length versus Time Curve

6.1.2. Sampling Effects on Trajectory Planning

The effects of discrete time approximation in Eqs. (3.5)-(3.7) are investigated in this section. Different servo time intervals are used for trajectory planning of the same straight line path as in Section 6.1.1. The smoothness constraints in Table 6.2 are used. It is expected that a large servo time interval may induce erroneous results due to the discretized approximation of the joint velocity, acceleration and jerk.

In Section 3.4.2, the existence of a straight line trajectory for a given path requires that the inverse kinematic solution from $\rho_1 \cdot \Delta\hat{\lambda}^1(k)$ be bounded by the smoothness constraints and that from $\rho_2 \cdot \Delta\hat{\lambda}^1(k)$ not be bounded by the smoothness constraints in the algorithm FW. The use of a large servo time interval may initially result in large approximated error in $\Delta\hat{\lambda}^1(k)$, which in turn induces a failure in satisfying the requirements for the existence of straight line trajectory from the search algorithms. Thus, the existence of the solution from the search algorithms is questionable if a large servo time interval is used to approximate the joint velocity, acceleration, and jerk.

In this simulation, the same values of ρ_1 and ρ_2 as in Section 6.1.1 were used for the algorithms FW and MFW, respectively. Figures 6.8 - 6.13 show the joint position, velocity, and acceleration trajectory for the given straight line path with a servo time period, 20 *msec*, where markers are used on the trajectory for every 60 *msec*. Again, most of joint trajectories are found to be constrained by the limits from the jerk bound and the given straight line requirement. Figure 6.11 shows that the acceleration of joint 4 reaches the maximum allowable limit for the acceleration and deceleration portion of the straight line trajectory.

For the same straight line path as simulated in Section 6.1.1, the following effects from using a different sampling period (20 *msec*) are found: (1) the total traveling time is reduced to 1.62 sec from 1.71 sec of Section 6.1.1. (2) the break point exists at 42^{th} sampling instant. (3) the Cartesian distance error at the final search point is found to be 0.242 *mm* with the maximum Euler angle error of 0.036 *degrees* (before relaxation). (4) the Cartesian distance error at the final search point is found to be 1.749 *mm* with the maximum Euler angle error of 0.258 *degrees* (after relaxation). (5) the required number of relaxation points is found to be 3. These can be compared with the contents in Table 6.3.

By increasing the servo time interval to 30 *msec*, it was found that the discrete time approximation results in a failure of applying the search algorithms. As a remedy of this case, the procedure ESLT in Section 3.4.2 can be applied.

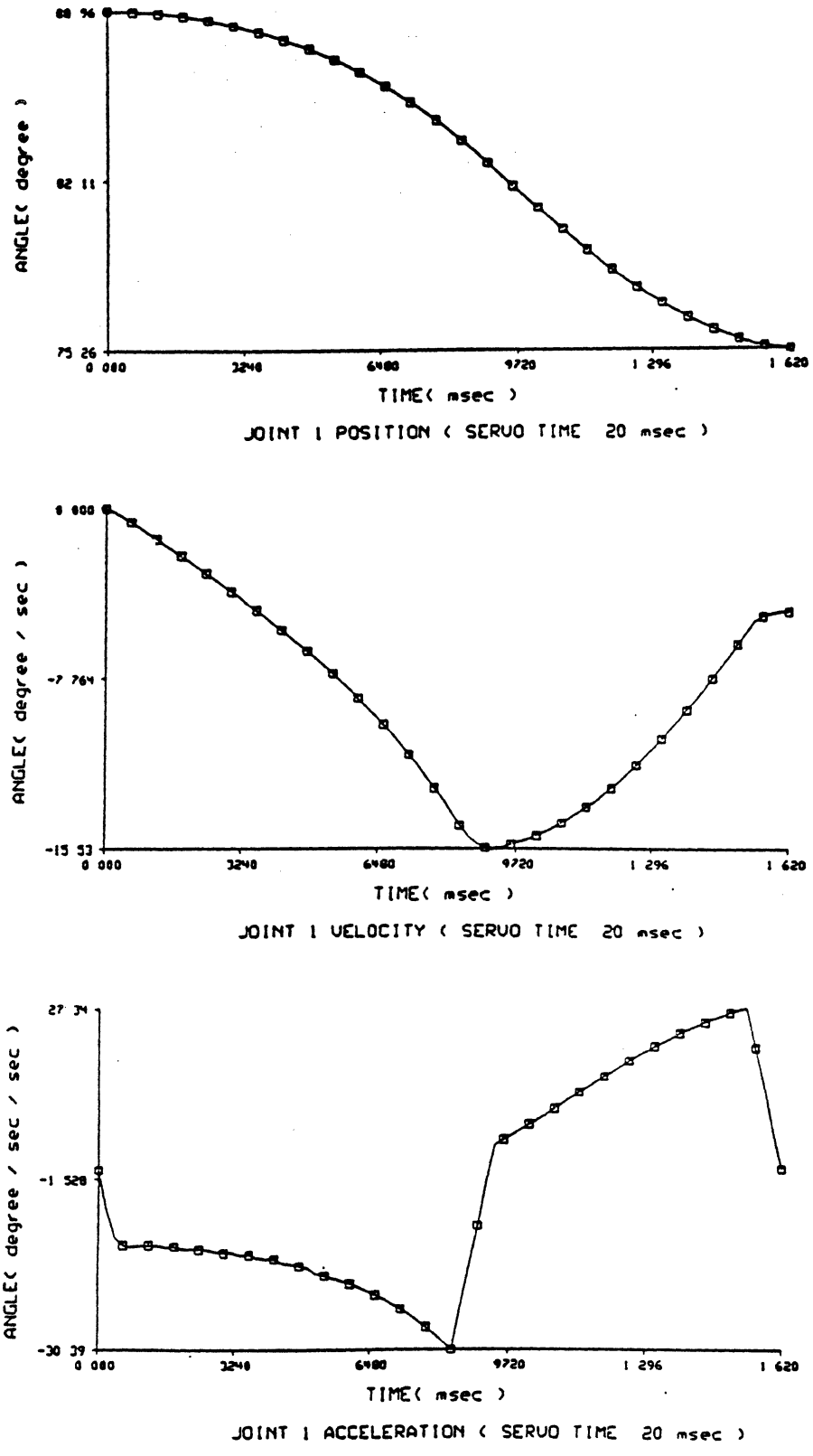


Figure 6.8 Sampling Effects on Joint 1 Trajectory

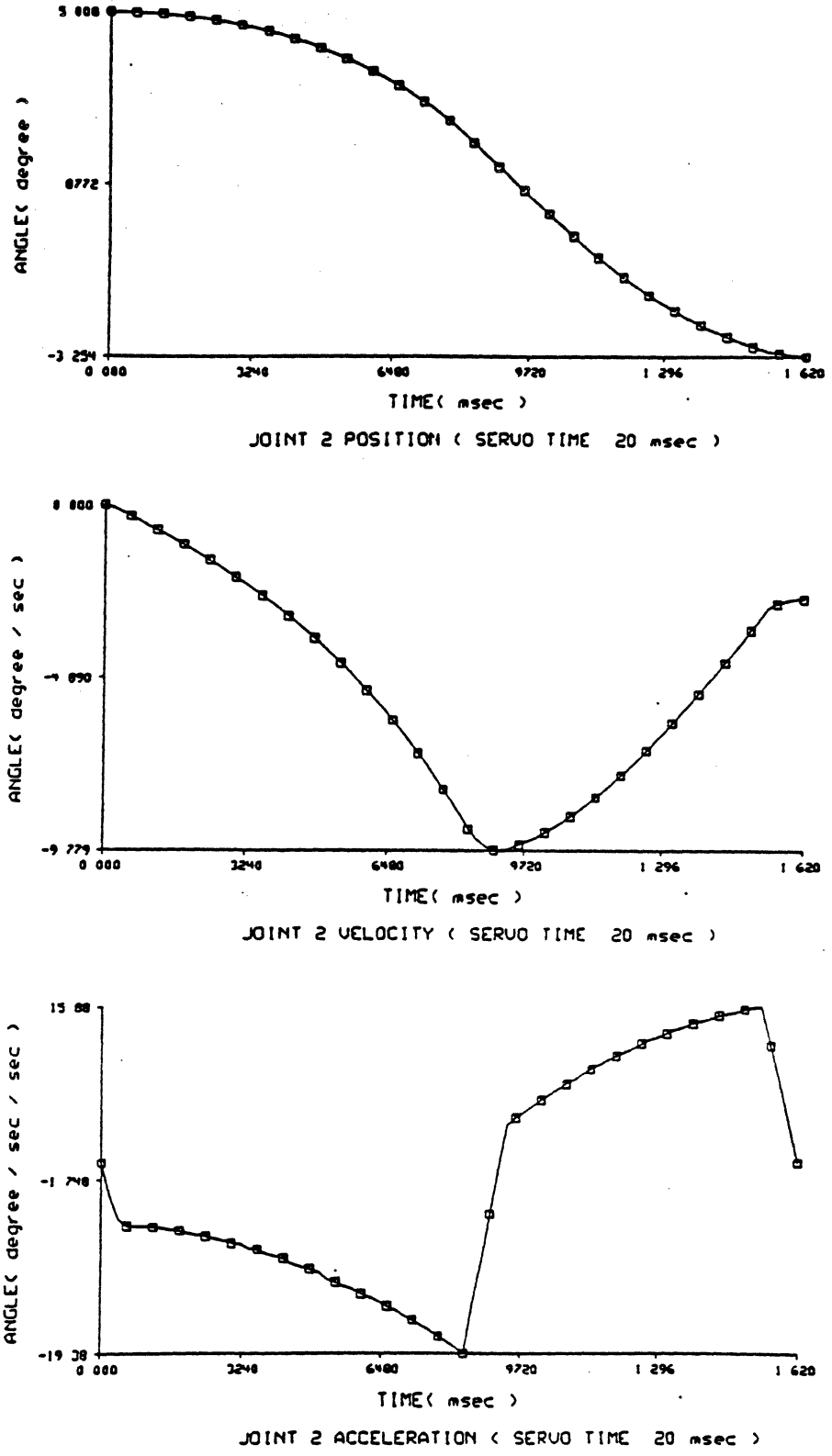


Figure 6.9 Sampling Effects on Joint 2 Trajectory

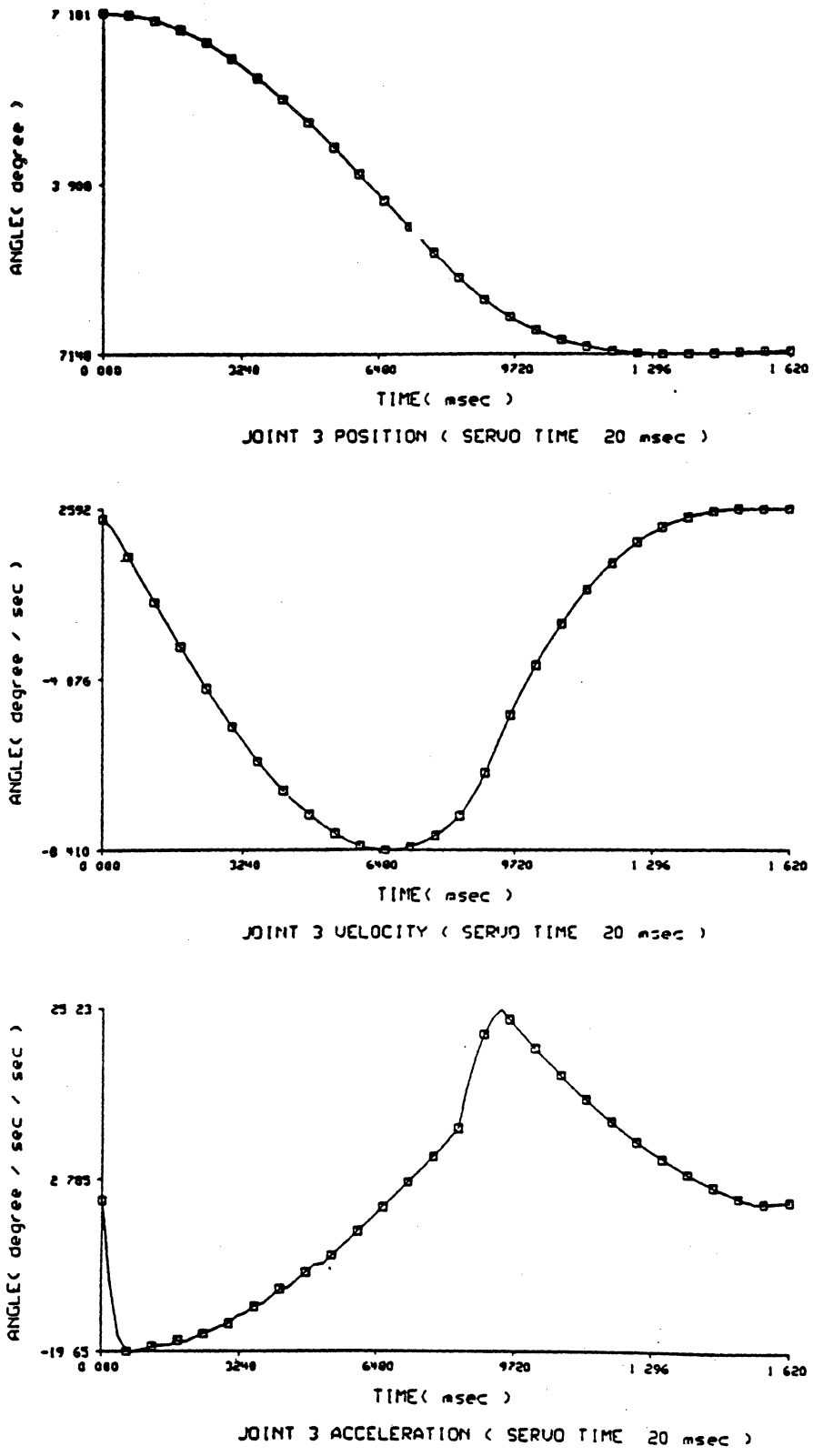


Figure 6.10 Sampling Effects on Joint 3 Trajectory

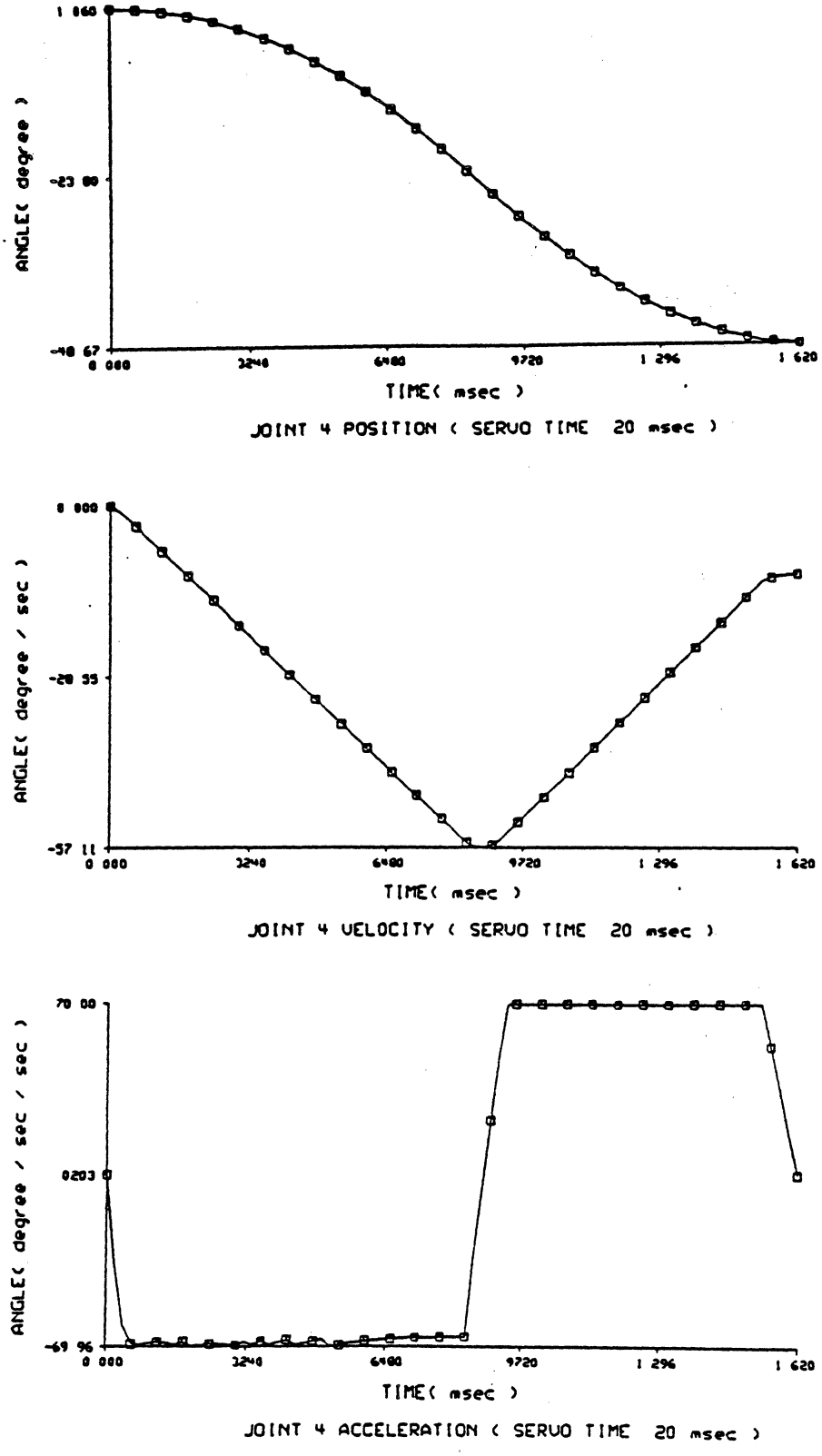


Figure 6.11 Sampling Effects on Joint 4 Trajectory

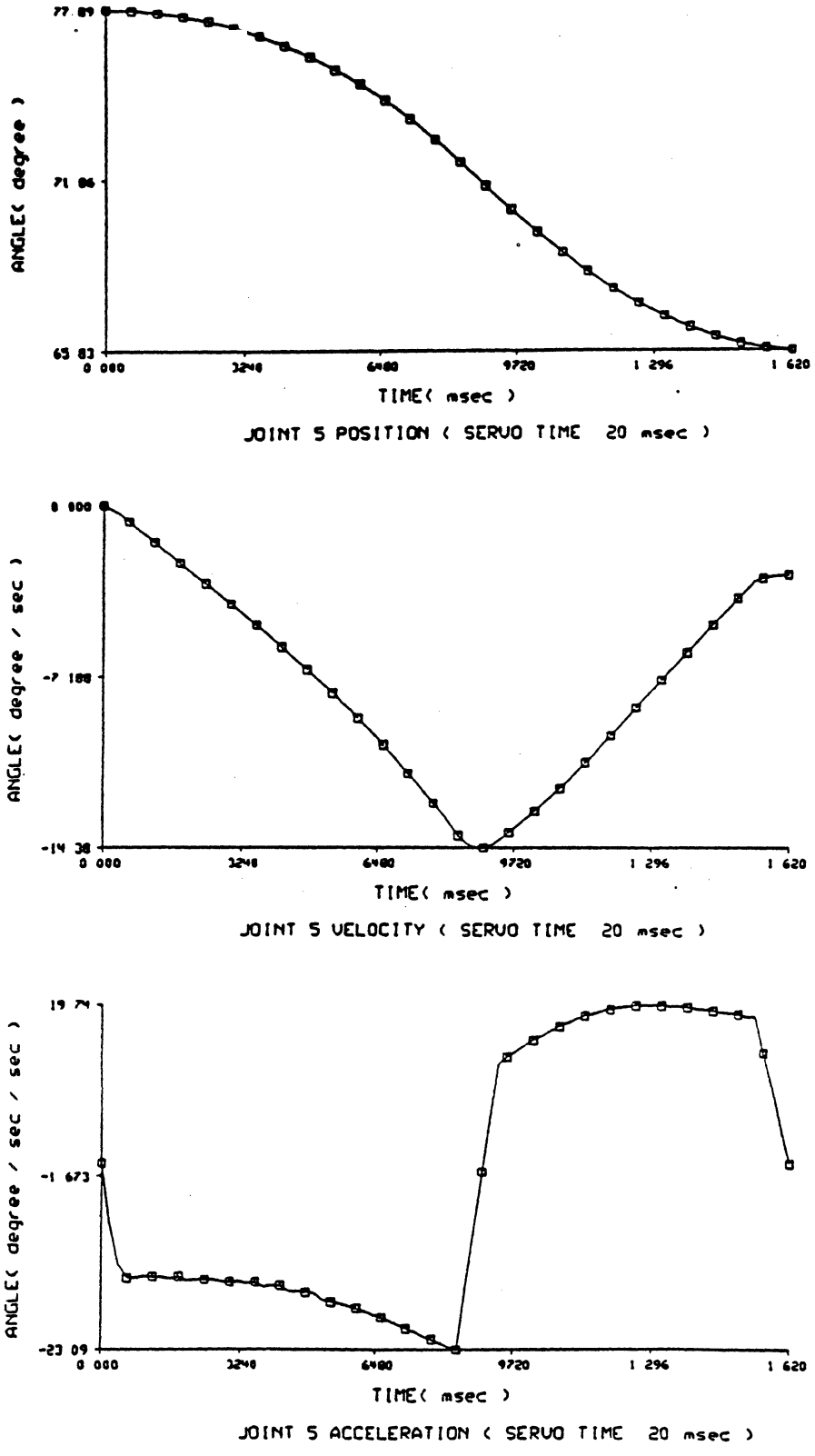


Figure 6.12 Sampling Effects on Joint 5 Trajectory

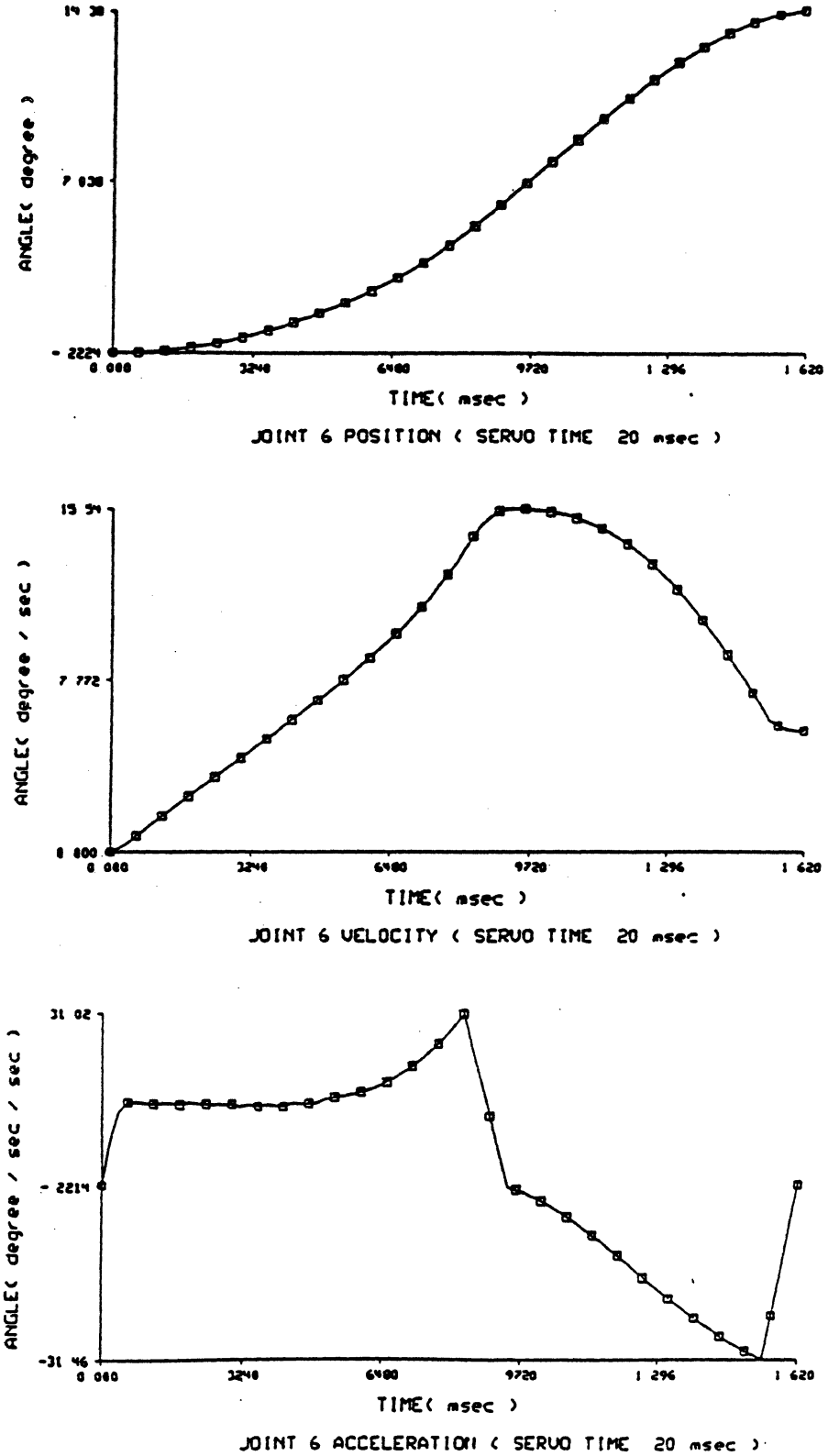


Figure 6.13 Sampling Effects on Joint 6 Trajectory

6.2. Existence of Straight Line Trajectory

This section describes the simulation results on the existence of the straight line trajectory. The procedure ESLT discussed in Section 3.4.2 is applied to a different path from Section 6.1, which must be planned by the procedure ESLT.

The path is shown schematically in Figure 6.14. The points A and F are the initial and the final point, respectively. The application of the algorithms FW and MFW to the path is found to be a failure. The initial estimate $\rho_1 \cdot \Delta\hat{\lambda}^1(k)$ did not satisfy the smoothness constraints during the execution of the algorithm FW. Thus we apply the procedure ESLT to the path. The point B is the final search point of the algorithm FW. From point B, the algorithm MFW is applied with point B as an initial location and F as a final target location. The algorithm MFW is stopped at a point C on the path. Hence, $\overline{AB}$ is an acceleration portion and $\overline{BC}$ is a deceleration portion of the straight line segment $\overline{AC}$.

From point C, the FW algorithm is applied again with point C as an initial location and F as a final target location. It is found that the algorithm FW is stopped by the stopping variable δ_1. Since the application of the algorithm is completed and stopped by δ_1, the algorithm BW and the procedure BR are applied to find a break point on the segment $\overline{CF}$. The break point is denoted by a point D. From the break point D, the algorithm MFW is applied until it is stopped at point E by the stopping variable δ_1. Thus, $\overline{CD}$ is the second acceleration portion and $\overline{DE}$ is the second deceleration portion of the straight line trajectory.

As a result of the application of the procedure ESLT to the given straight line path, the resultant trajectory is composed of two acceleration portions and two deceleration portions. The points B, C and D are the break points in this simula-

tion. The detailed data for these points are shown in Table 6.5. Also, the traveled length versus servo time curve is drawn in Figure 6.15 without the relaxation process. The joint position, velocity and acceleration trajectories are shown in Figures 6.16-6.21, where markers are used for every $50msec$.

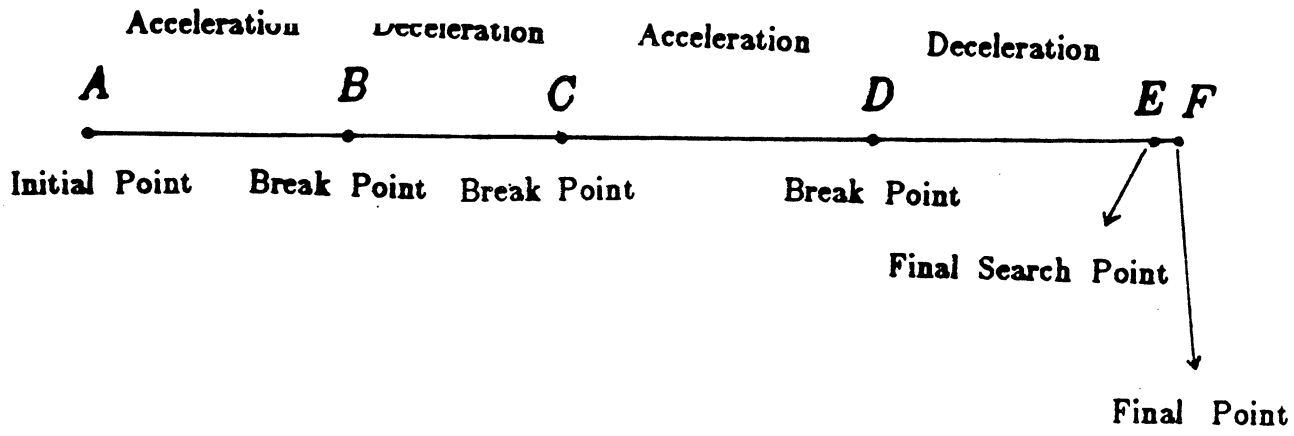


Figure 6.14 Straight Line Path for the Procedure ESLT

Point	Search Point	p_x or Joint 1	p_y or Joint 2	p_z or Joint 3	α or Joint 4	β or Joint 5	γ or Joint 6
A	p_A, Φ_A	-0.140000	0.500000	0.290000	30.000000	90.000000	90.000000
	$\dot{q}_A$	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	$\ddot{q}_A$	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
B	p_B, Φ_B	-0.118803	0.467905	0.288803	49.998454	85.349221	84.650783
	$\dot{q}_B$	-4.447136	3.314219	-16.853863	-41.101005	33.945357	-5.582826
	$\ddot{q}_B$	-13.799862	14.172843	-24.810662	-99.258118	42.961890	-36.575863
C	p_C, Φ_C	-0.100298	0.440448	0.250298	49.850771	89.925387	90.074861
	$\dot{q}_C$	-3.535689	4.707196	-8.913738	-22.943288	16.281314	-6.610428
	$\ddot{q}_C$	8.050623	-14.121416	28.345285	79.009439	-57.032080	13.185199
D	p_D, Φ_D	-0.081583	0.427300	0.241583	54.203350	72.101676	77.898321
	$\dot{q}_D$	-5.616465	8.529499	-11.944806	-31.332810	20.110128	-10.877418
	$\ddot{q}_D$	-17.485284	15.197171	-25.188323	-88.984911	30.667549	-34.987844
E	p_g, Φ_g	-0.078208	0.404312	0.228208	61.896049	75.949029	74.051968
	$\dot{q}_g$	-3.110468	2.938889	-5.047681	-13.445910	6.882291	-5.888477
	$\ddot{q}_g$	14.411699	-14.480001	25.391228	67.818873	-37.586040	27.781794
F	p_F, Φ_F	-0.078000	0.404000	0.228000	62.000000	76.000000	74.000000
	$\dot{q}_F$	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
	$\ddot{q}_F$	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000

Units: p is in meters, Φ is in degrees, $\dot{q}$ is in deg/sec, and $\ddot{q}$ is in deg/sec^2

Table 6.5 Break Point Information from the Procedure ESLT

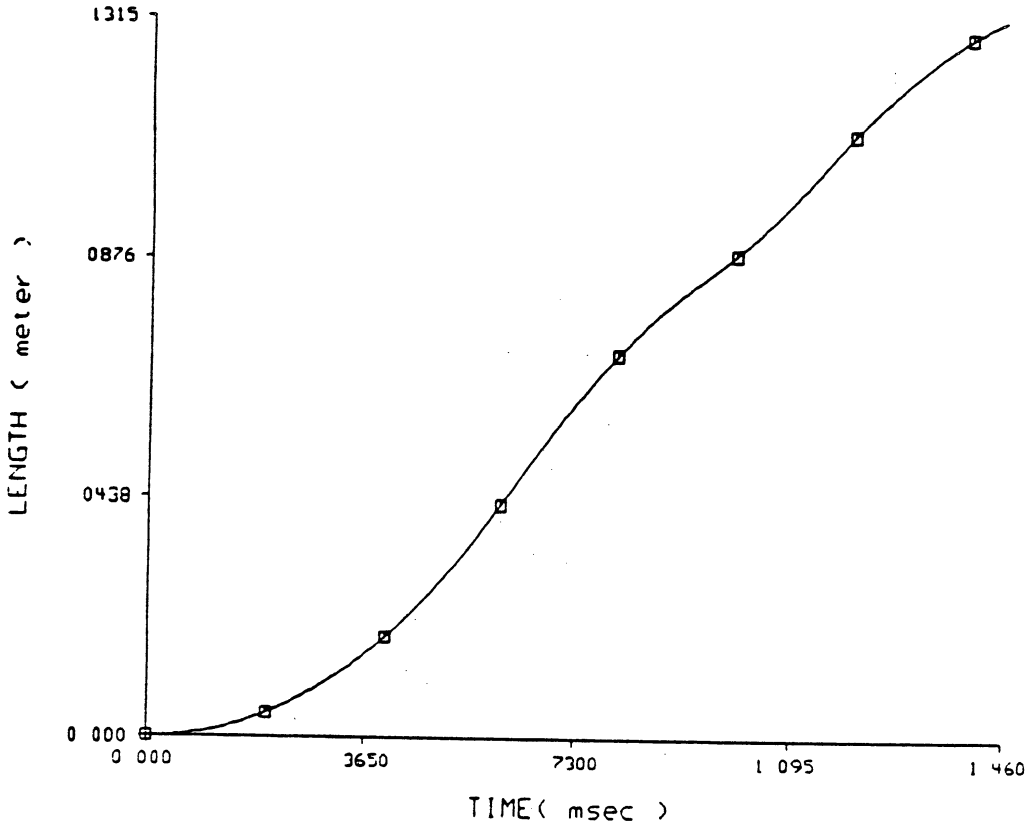


Figure 6.15 Traveled Length versus Servo Time Curve

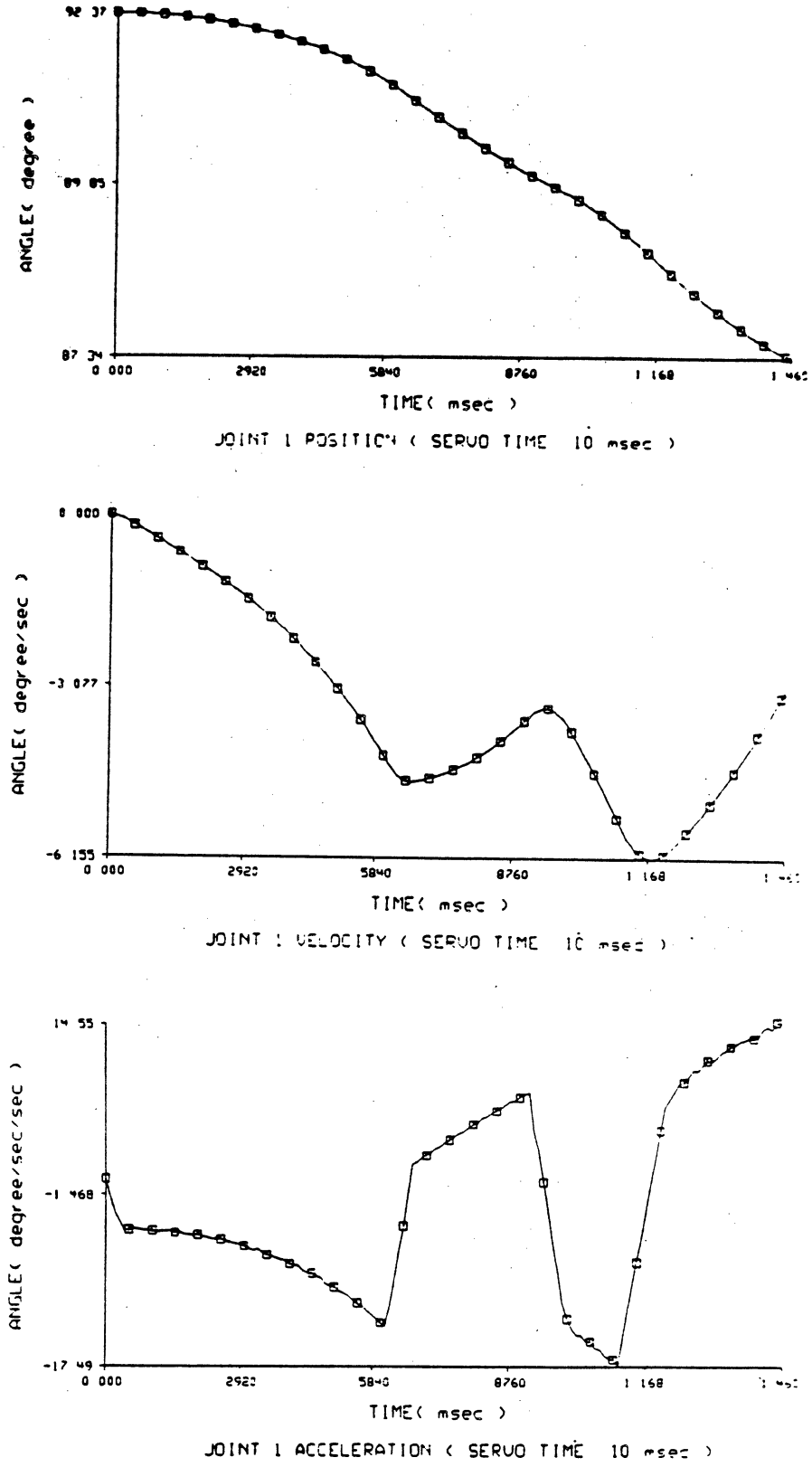


Figure 6.16 Joint 1 Trajectory from the Procedure ESLT

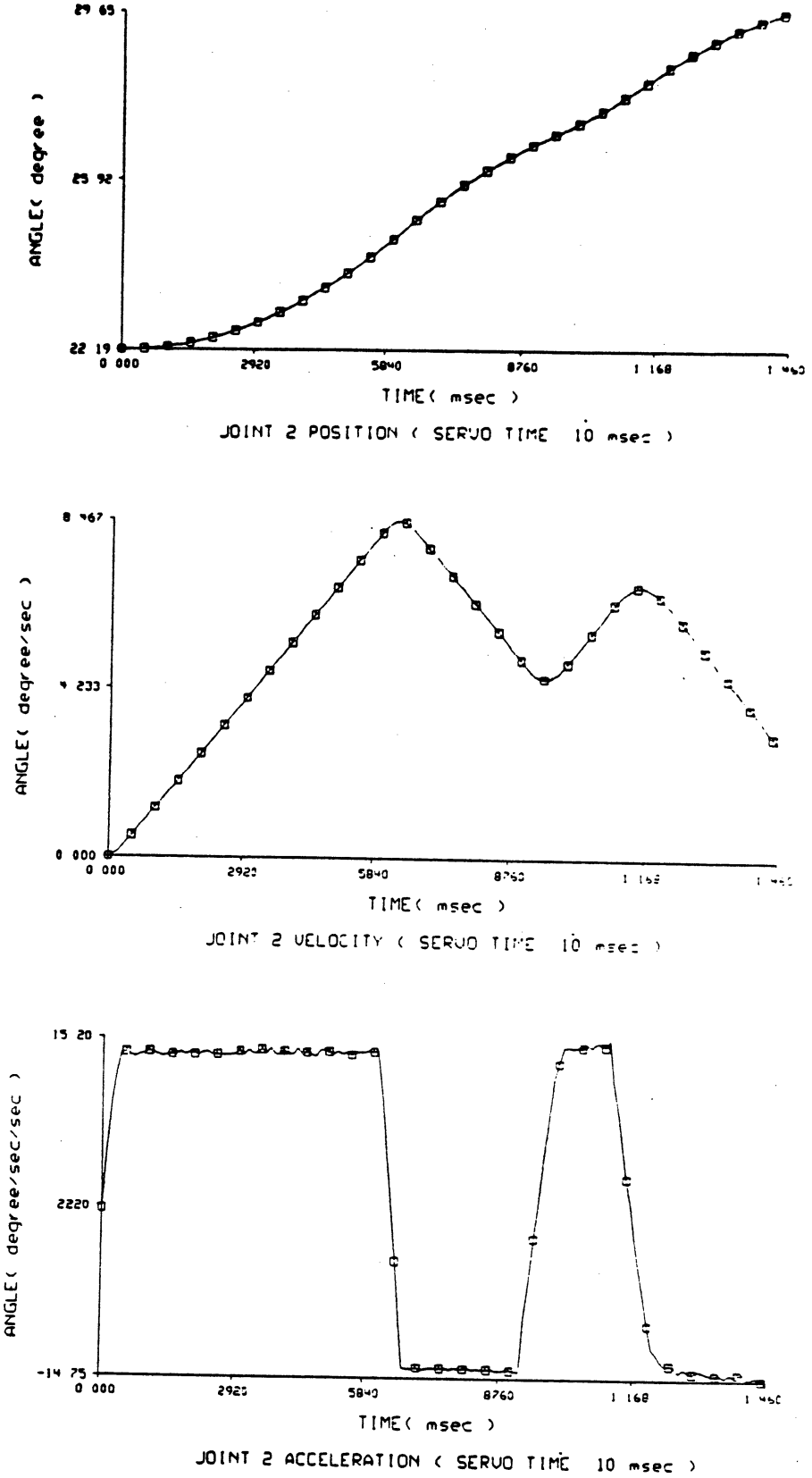


Figure 6.17 Joint 2 Trajectory from the Procedure ESLT

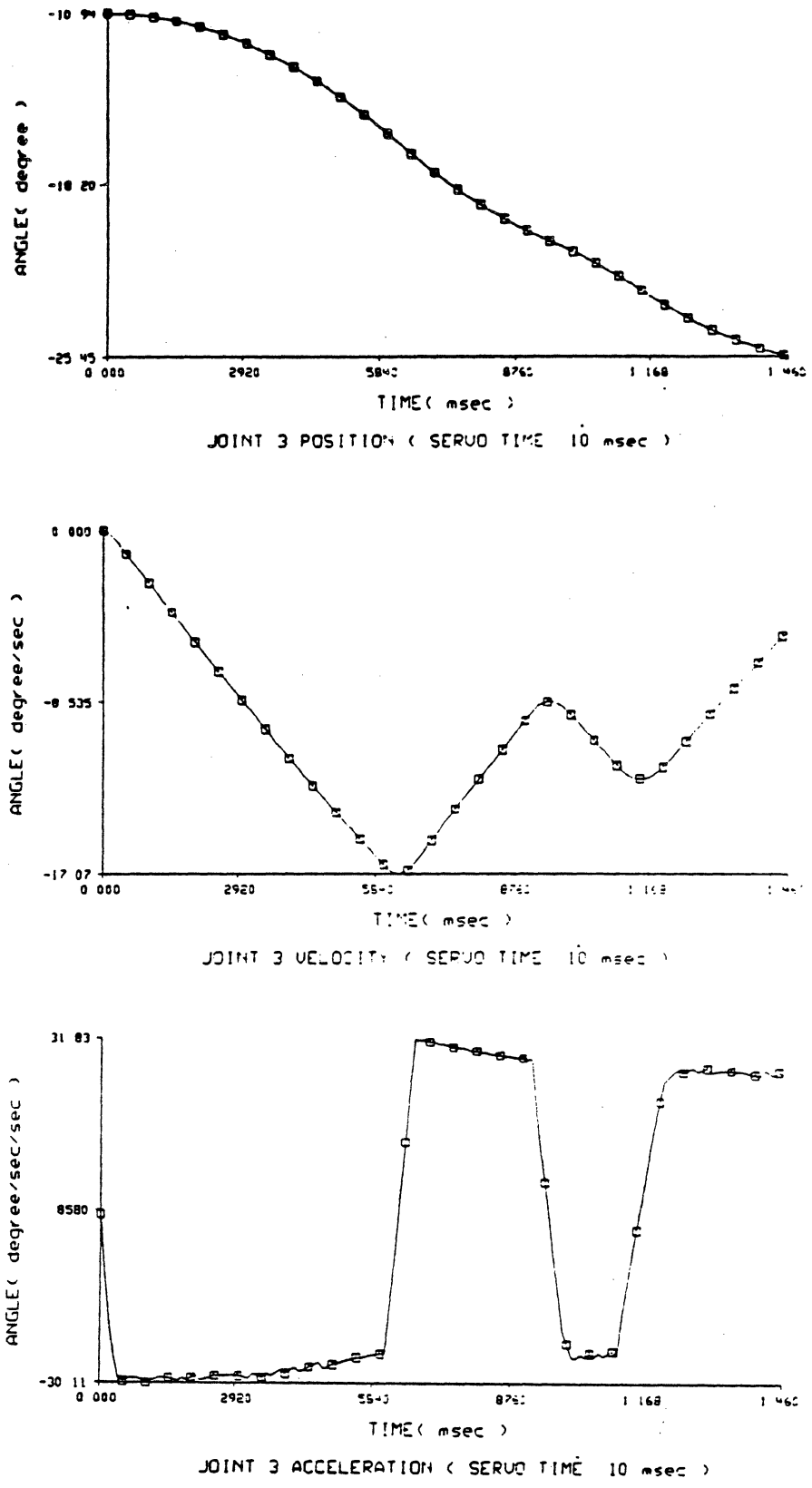


Figure 6.18 Joint 3 Trajectory from the Procedure ESLT

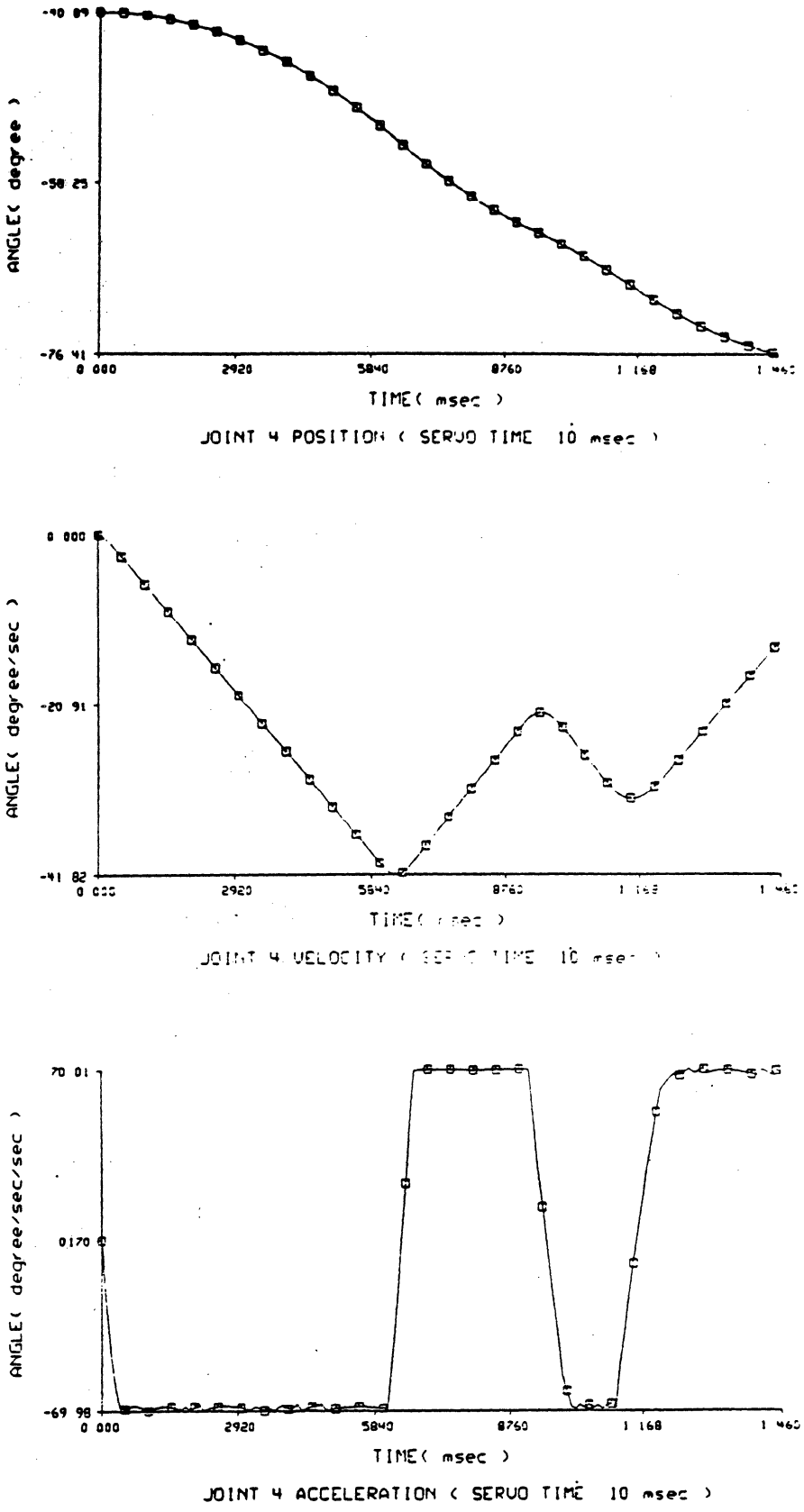


Figure 6.19 Joint 4 Trajectory from the Procedure ESLT

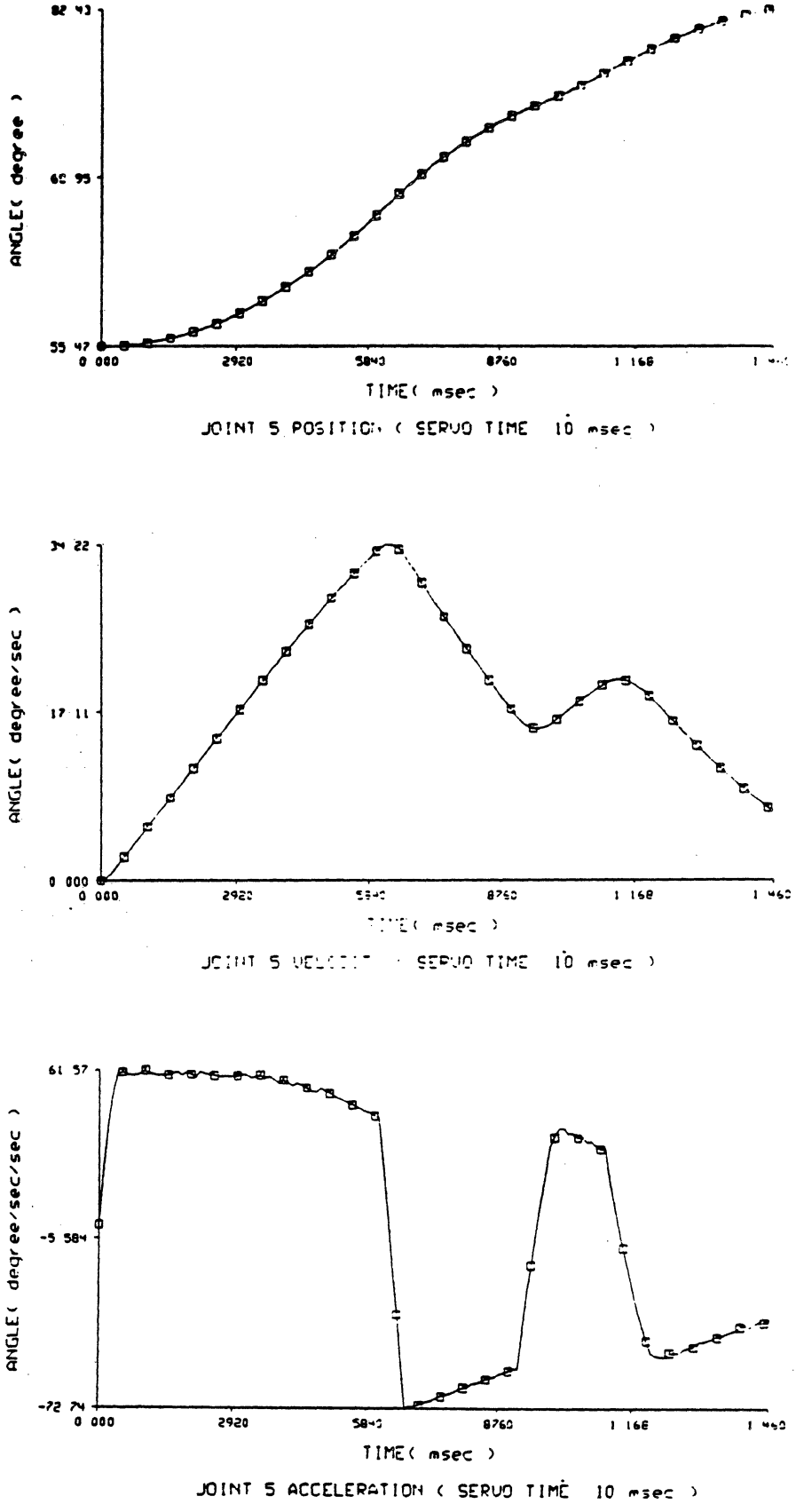


Figure 6.20 Joint 5 Trajectory from the Procedure ESLT

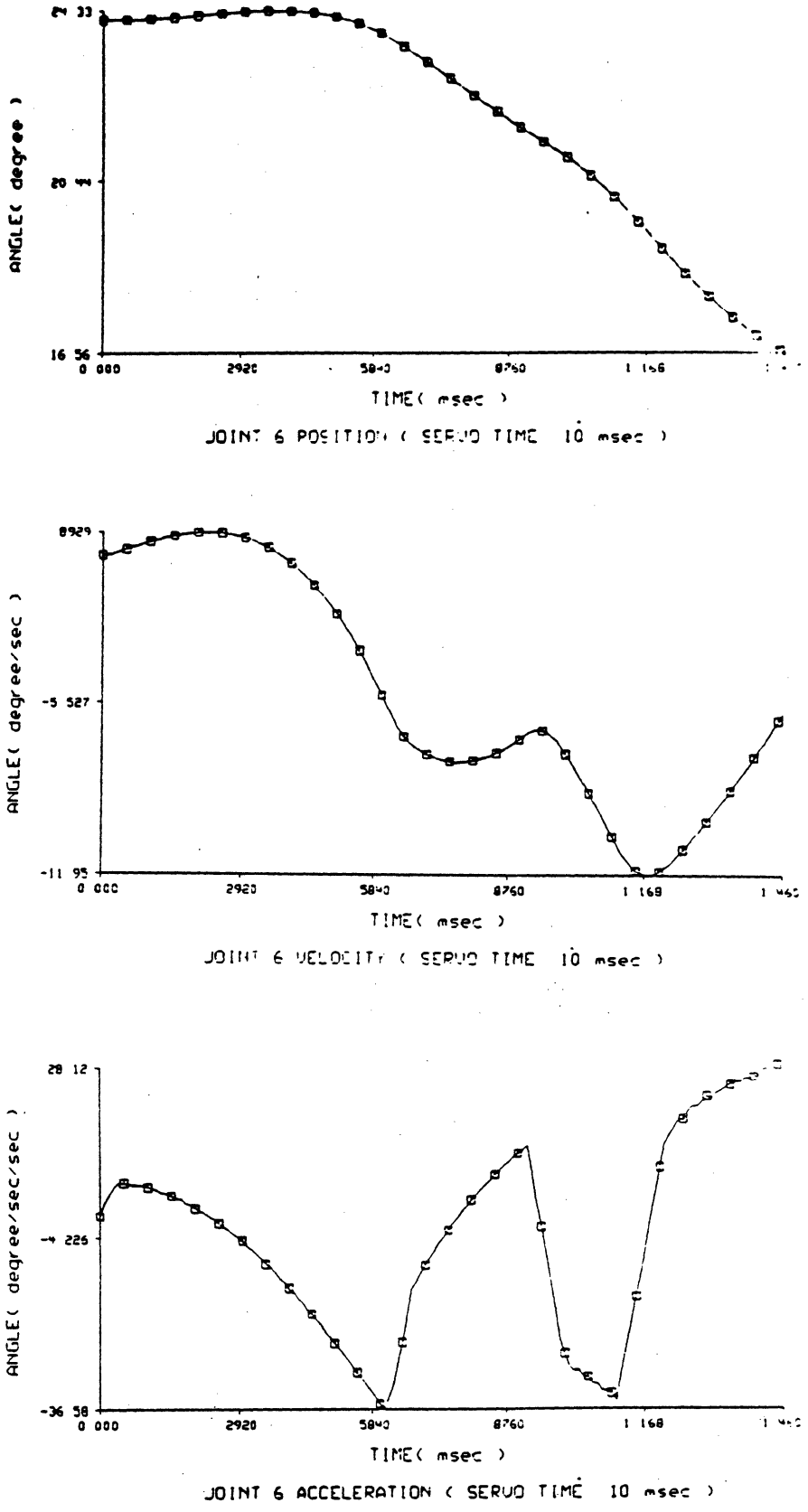


Figure 6.21 Joint 6 Trajectory from the Procedure ESLT

6.3. Trajectory Planning of Connected Straight Line Segments

The simulation of two connected straight line segments is described in this section. To follow the connected straight line segment faithfully, the robot must stop near the intersection point of the connected straight line segment for changing the traveling direction to the final desired point. As discussed in Section 3.4.3, if we adopt the cubic splined relaxation process, the relaxation portion of the straight line trajectory can meet the desired boundary conditions at the final location [34]. That is, the intersection point can be reached exactly and the trajectory planning of a connected straight line segment can be performed as two independent straight line paths.

Here, we present a simulation result from the discrete time relaxation process and show that there is an error near the intersection point of the connected straight line segments. In Figure 6.22, the connected straight line segments are shown schematically. The locations of various points are listed in Table 6.6, where break points, final points from the relaxation process, and initial and final points for the deviated straight line path are tabulated with corresponding servo time instants. The second straight line segment must be modified slightly because of the error in planning the first segment. The Cartesian distance error between the desired corner point and the actual corner point was found to be 0.274 *mm* with 0.04 *degrees* for the maximum Euler angle error. The collision-free path must tolerate this deviation for the collision-free motion planning.

The joint position, velocity, and acceleration trajectories are shown in Figures 6.23 - 6.28, where markers are used on the trajectory for every 50 *msec* and arrows are shown at 1.71 sec for stopping instant at the corner point. Most of joint trajec-

tories are found to be constrained by the limits from the jerk bound and the given straight line requirement. However, Figure 6.26 shows that the acceleration of joint 4 reaches the maximum allowable limit for the acceleration and deceleration portion of the connected straight line segments.

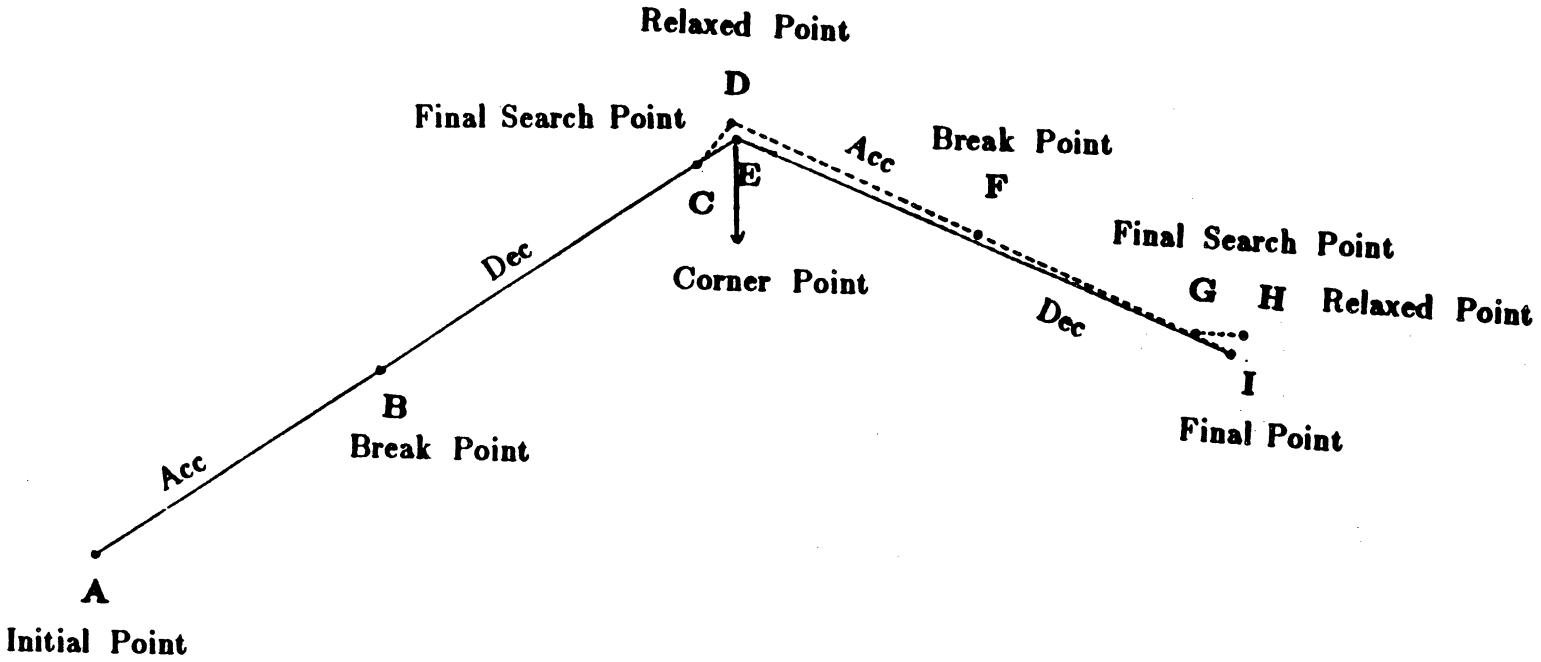


Figure 6.22 Simulated Connected Straight Line Segments

Point	Interpretation	p_x (m)	p_y (m)	p_z (m)	α (degree)	β (degree)	γ (degree)	Travel Time (sec)
A	Initial Point	-0.140000	0.560000	0.390000	0.000000	90.000000	90.000000	0
B	Break Point(ACC-DEC)	-0.073779	0.503239	0.432570	14.190130	75.809888	75.809888	0.84
C	Final Search Point	-0.000028	0.440024	0.479982	29.994048	60.005951	60.005951	1.66
D	Relaxed Point	0.000187	0.439840	0.480120	30.040067	59.959956	59.959956	1.71
E	Corner Point	0.000000	0.440000	0.480000	30.000000	60.000000	60.000000	
F	Break Point(ACC-DEC)	0.000097	0.391912	0.480062	37.221604	50.378154	55.178670	2.30
G	Final Search Point	0.000000	0.340036	0.480000	44.994643	40.007147	50.003566	2.81
H	Relaxed Point	-0.000001	0.339297	0.479999	45.105306	39.859365	49.929683	2.84
I	Final Point	0.000000	0.340000	0.480000	45.000000	40.000000	50.000000	

Table 6.6 Connected Straight Line Segment Location Information

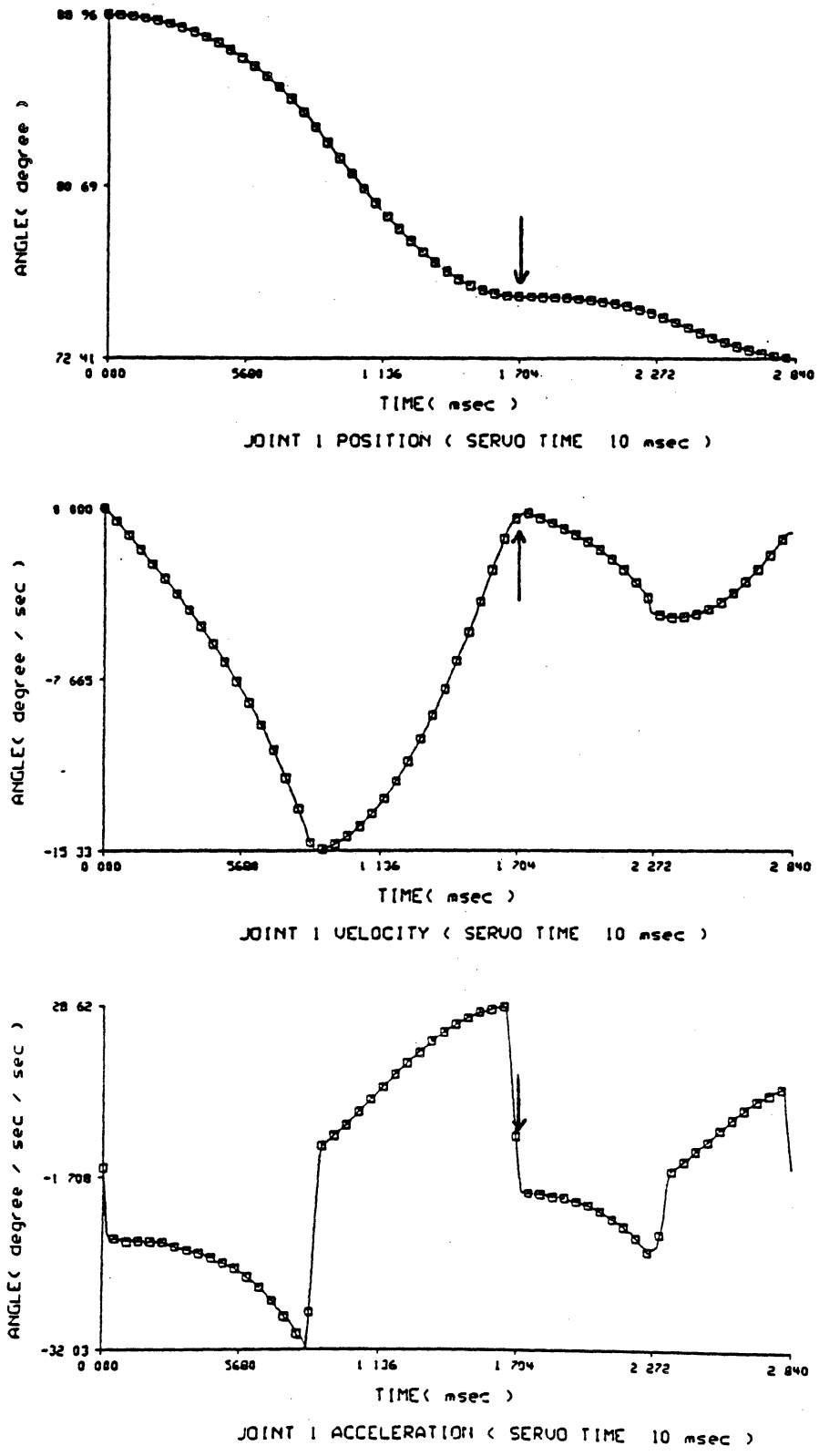


Figure 6.23 Joint 1 Trajectory of Connected Straight Line Segments

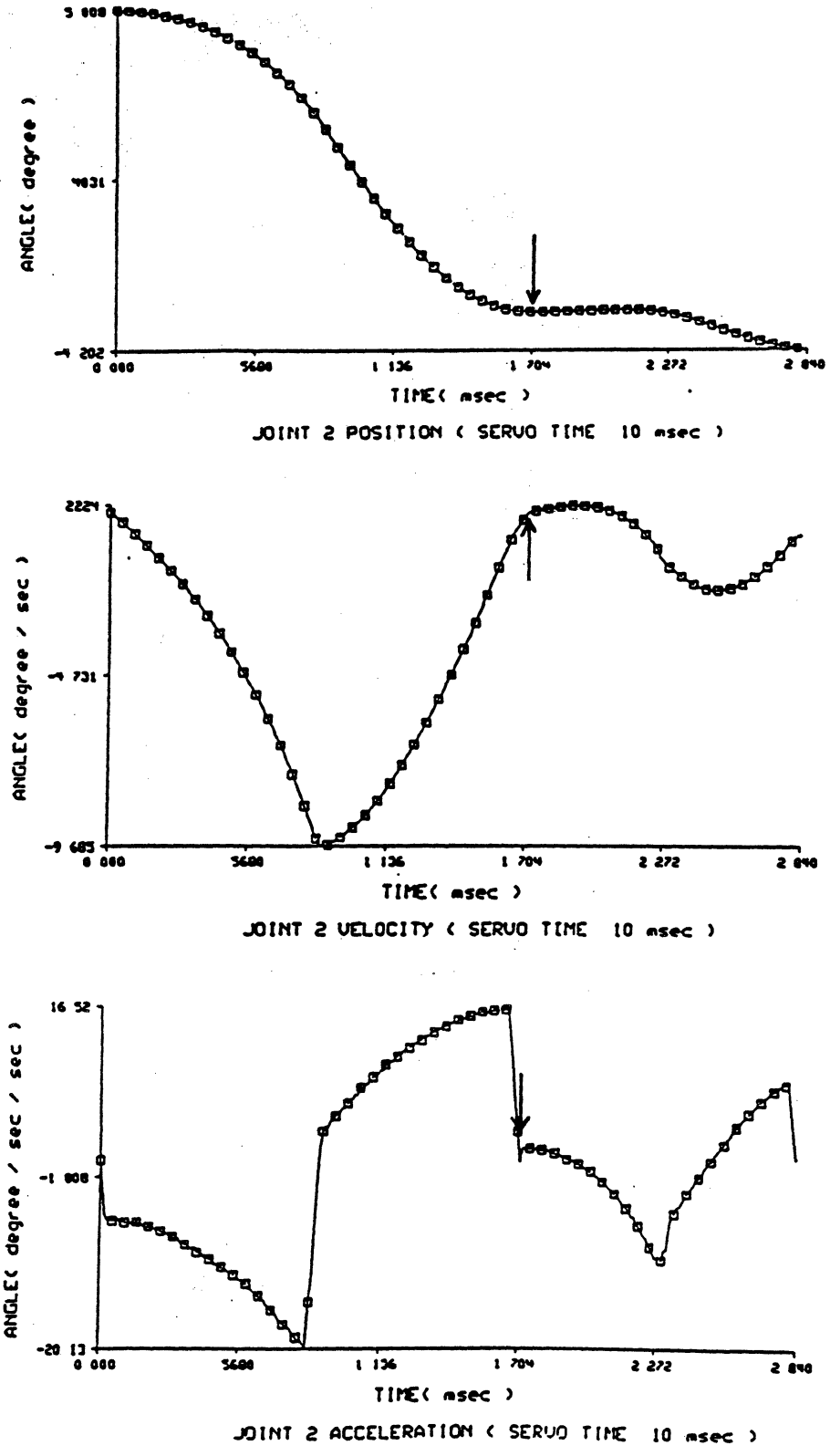


Figure 6.24 Joint 2 Trajectory of Connected Straight Line Segments

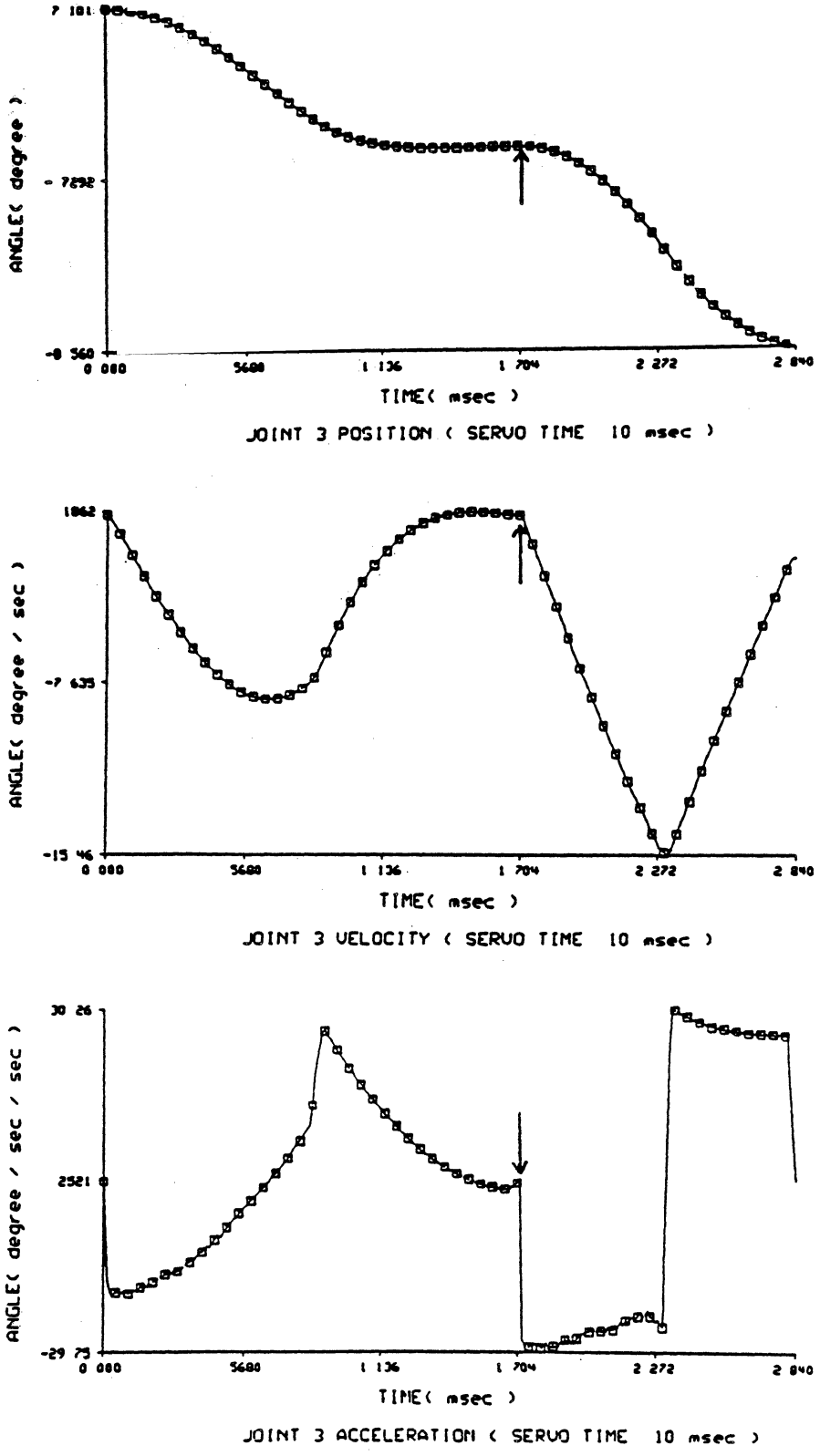


Figure 6.25 Joint 3 Trajectory of Connected Straight Line Segments

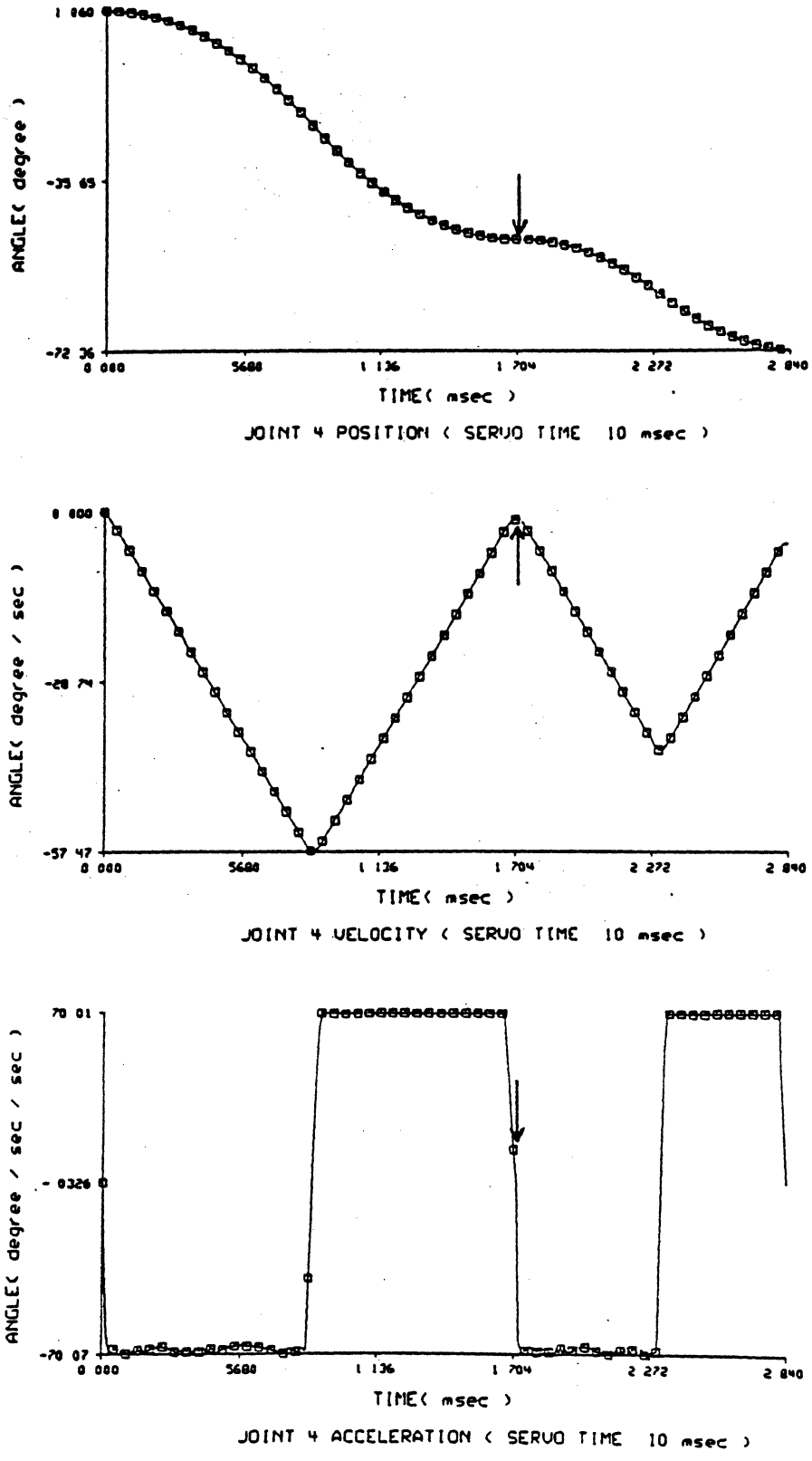


Figure 6.26 Joint 4 Trajectory of Connected Straight Line Segments

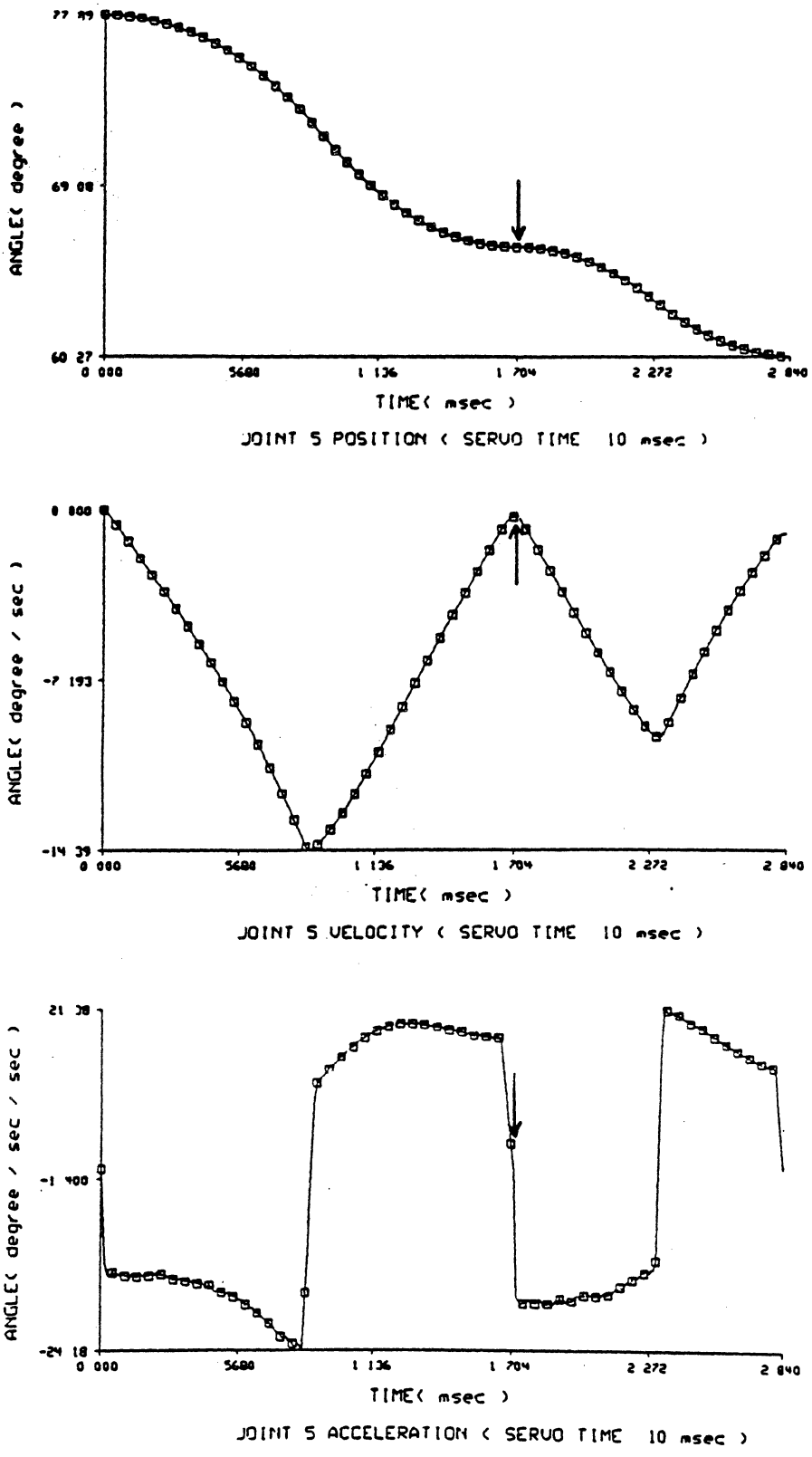


Figure 6.27 Joint 5 Trajectory of Connected Straight Line Segments

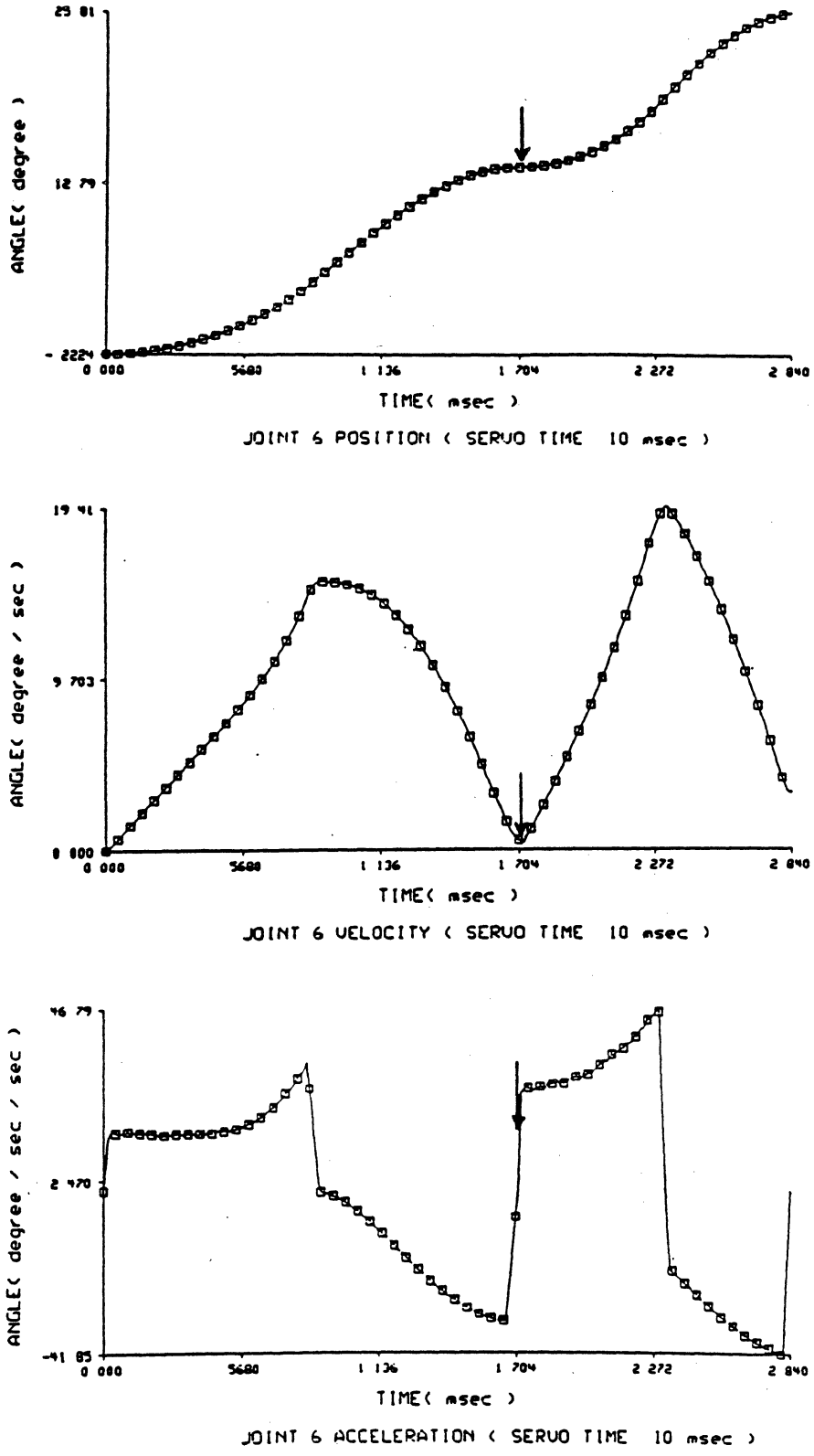


Figure 6.28 Joint 6 Trajectory of Connected Straight Line Segments

6.4. Time Scheduling of a Straight Line Trajectory

The time scheduling of a straight line trajectory is performed in this section. The simulation was done by applying the procedure TS to the straight line trajectory.

We assume that a collision map is given as shown in Figure 6.29, where the collision box covers the length from 4.9 *cm* to 10.8 *cm* during the time interval from 0.42 *sec.* to 0.74 *sec.*. That is, l_s, the collision starting length, is assumed to be 4.9 *cm* and l_e, the collision ending length, is assumed to be 10.8 *cm*. From the collision map, we have two straight line segments; one ranges from the initial point to the collision starting point, while the other ranges from the collision starting point to the desired final point. The first segment was planned to constitute an acceleration portion and a deceleration portion and was found able to avoid the collision box. The second segment was planned by applying the algorithms FW and MFW with the initial estimate $\Delta\hat{\lambda}^1(k)$, which can be obtained from the last trajectory set point information of the first straight line segment.

Figure 6.30 shows the collision map, where the traveled length versus servo time curve avoids the collision box. Figure 6.31 and Table 6.7 show the details of the simulation results for the time scheduling of the straight line trajectory. The total traveling time without the relaxation process was found to be 1.66 *sec.* from Section 6.1.1, while the total traveling time after time scheduling was found to be 1.97 sec. Detailed joint position, velocity, and acceleration trajectories are shown in Figures 6.32 - 6.37, where markers are used on the trajectory for every 50 *msec*.

We investigate the effect on the final arrival time when speed reduction of the robot motion is performed for avoiding the collision region in the collision map. As

discussed in Section 4.4.1, when the speed reduction is performed on the segment of length l_s, the traveling time of the segment of length l_s from speed reduction must be equal to or greater than the traveling time from the original trajectory plus the required time delay for avoiding the collision region. From the location of length l_s to the final location, the traveling time from the original trajectory is shorter than that from the trajectory after speed reduction because the robot on the original trajectory follows a maximum accelerating and decelerating trajectory.

If we assume that the robot motion can be delayed at the initial location, then the required time delay can be computed from the collision map and the trajectory information. In Figure 6.29, the time k_1 was found to be 0.60 sec from the original trajectory information, thus the required time delay at the initial location will be $(0.74 - k_1)$ sec or 0.14 sec. Then, the final arrival time will be 1.80 sec which is shorter than that of the curve by speed reduction (1.97 sec).

Now, we investigate the effects on the final arrival time when the speed reduction of the robot motion is performed well before the robot reaches the collision starting location. Figure 6.38 is the same figure as for the previous example (Figures 6.29 and 6.30), where we denote the original traveled length versus servo time curve as OM and the time scheduled traveled length versus servo time curve as OM^{CD}. If we consider two possible collision regions C and D as shown in Figure 6.38, the curve OM^{CD} can be regarded as a curve which can be obtained by performing the time scheduling of the original trajectory well before the robot reaches the collision starting location for the collision regions C and D, respectively.

When we consider the collision region C, which the curve OM^{CD} can avoid, time k_s^C was found to be 0.70 sec and time k_e^C to be 0.97 sec with corresponding

lengths of 6.69 *cm* and 9.11 *cm*, respectively. These times and lengths can be obtained from the trajectory information (trajectory set points printed out from simulation) corresponding to curve OM. If the time delay is allowed for robot 2 to avoid the collision region C, the required time delay will be 0.27 sec, which yields 1.93 sec as the final arrival time. We notice that this final arrival time (1.93 sec) from the time delay is shorter than that from the speed reduction (1.97 sec). This corresponds to the curve OM^A as shown in Figure 4.13.

When we consider another collision region D, which the curve OM^{CD} can avoid, time k_s^D was found to be 0.99 sec and time k_e^D to be 1.36 sec with corresponding lengths of 13.29 *cm* and 15.32 *cm*, respectively. These times and lengths can be obtained from the trajectory information corresponding to curve OM^{CD}. If the time delay is allowed for robot 2 to avoid the collision region D, the required time delay will be 0.37 sec, which yields 2.03 sec as the final arrival time. We notice that this final arrival time (2.03 sec) from the time delay is longer than that from the speed reduction (1.97 sec). Therefore, we conclude the followings from discussions in Section 4.4.1. and this simulation:

(1) If speed reduction of the robot 2 motion is performed on the segment of length l_s for avoiding the collision region, then the final arrival time from pure time delay of the robot 2 motion can be shorter than that from speed reduction of the robot 2 motion. (See the collision maps in Figures 6.29-30.)

(2) If speed reduction of the robot 2 motion is performed well before robot 2 reaches the collision starting length l_s for avoiding the collision region, then the final arrival time from pure time delay of the robot 2 motion can be shorter (See the collision map with collision region C in Figure 6.38.) or

longer (See the collision map with collision region D in Figure 6.38.) than that from speed reduction of the robot 2 motion.

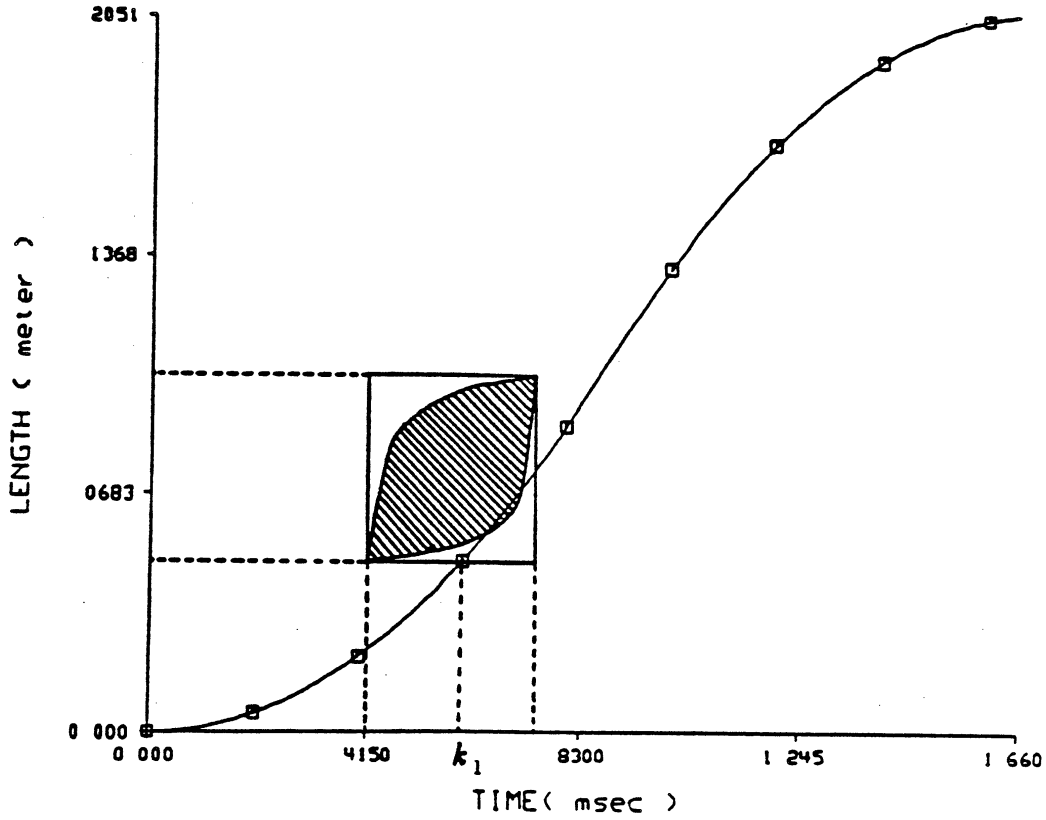


Figure 6.29 Collision Map with Possible Collision

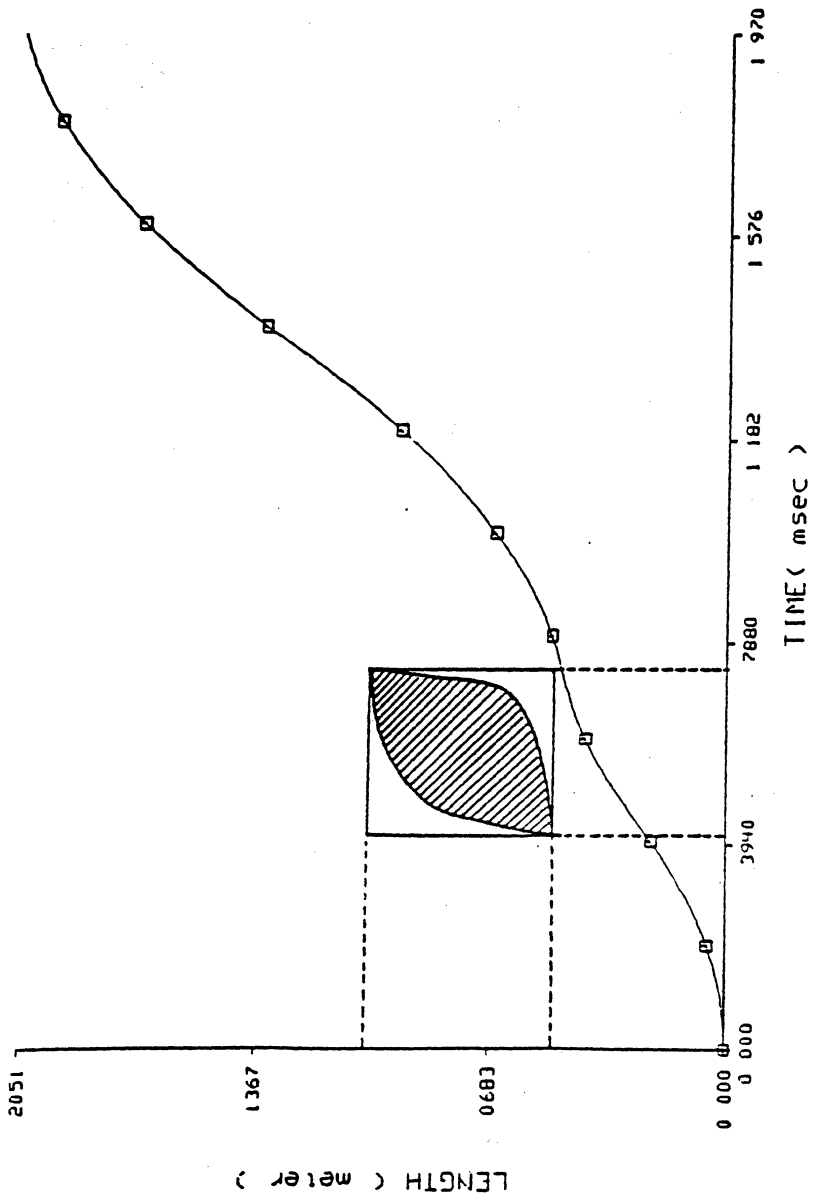


Figure 6.30 Time Scheduled Collision Map

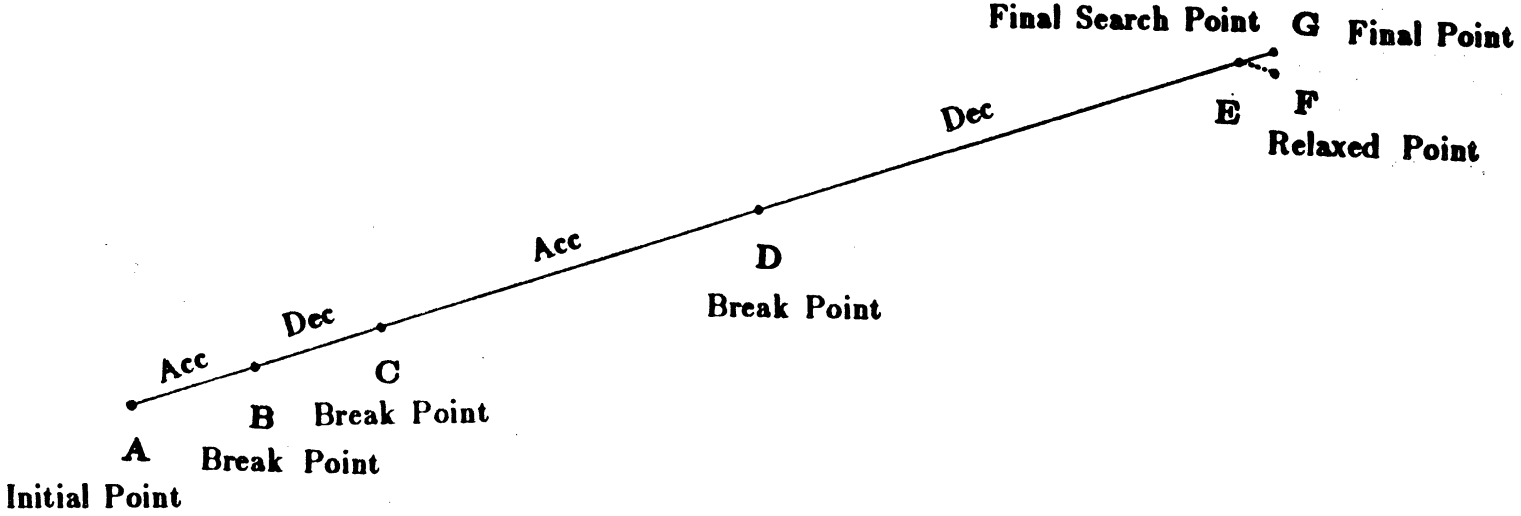


Figure 6.31 Simulated Straight Line Segment

Point	Interpretation	P_x (m)	P_y (m)	P_z (m)	α (degree)	β (degree)	γ (degree)	Travel Time (sec)
A	Initial Point	-0.140000	0.560000	0.390000	0.000000	90.000000	90.000000	0
B	Break Point(ACC-DEC)	-0.123641	0.545978	0.400516	3.505483	86.494517	86.494517	0.42
C	Break Point(DEC-ACC)	-0.107964	0.532540	0.410594	6.864926	83.135075	83.135075	0.71
D	Break Point(ACC-DEC)	-0.056955	0.488818	0.443386	17.795375	72.204634	72.204634	1.34
E	Final Search Point	-0.000046	0.440039	0.479971	29.990182	60.009817	60.009817	1.97
F	Relaxed Point	0.001157	0.439006	0.480745	30.247780	59.751724	59.751465	2.02
G	Final Point	0.000000	0.440000	0.480000	30.000000	60.000000	60.000000	

Table 6.7 Time Scheduled Trajectory Location Information

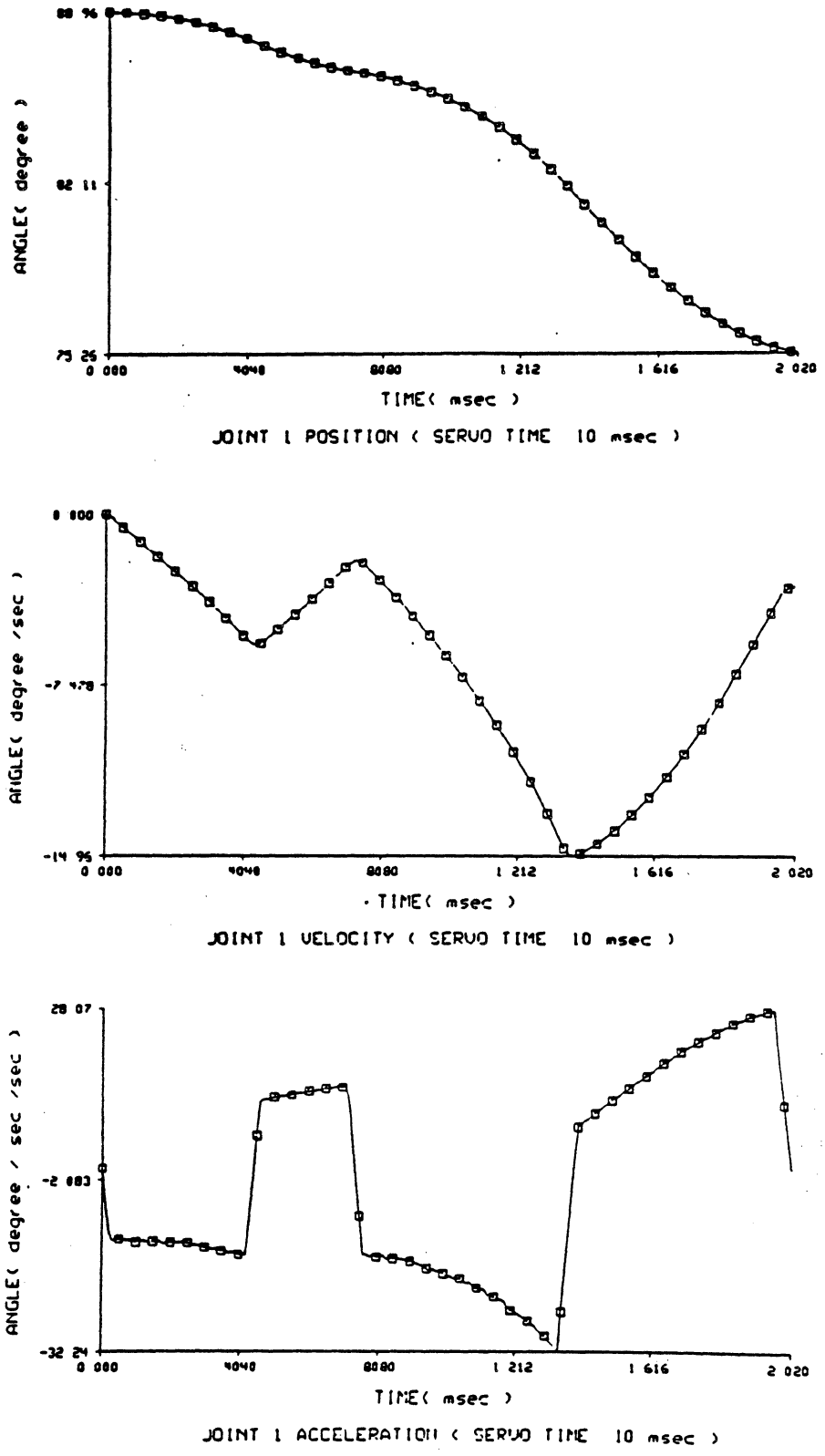
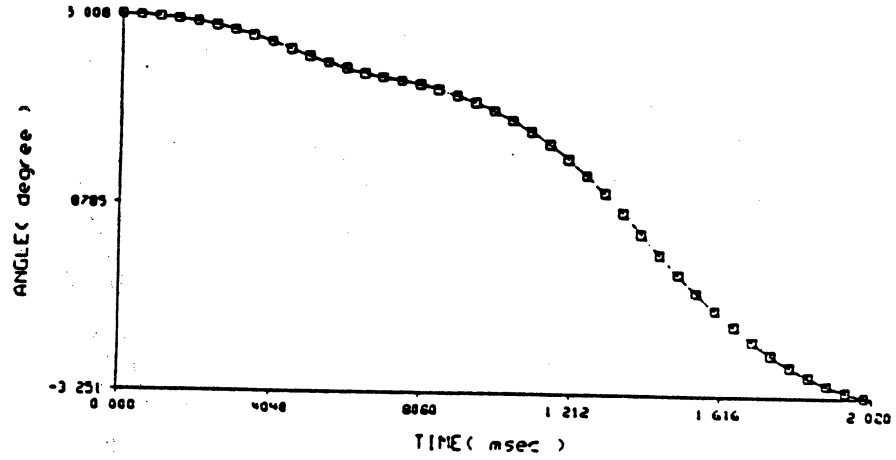
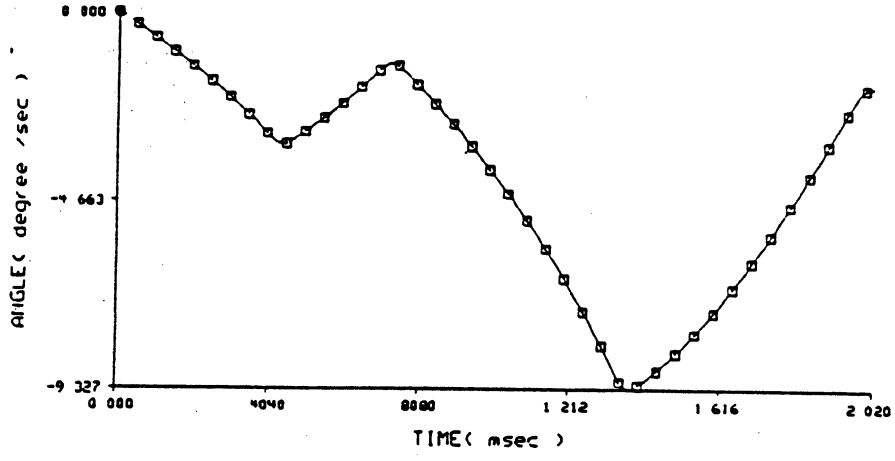


Figure 6.32 Time Scheduled Joint 1 Trajectory





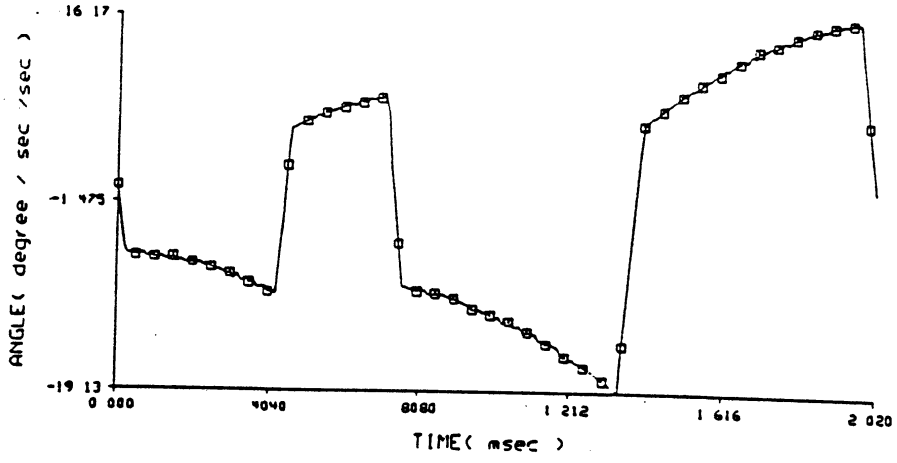


Figure 6.33 Time Scheduled Joint 2 Trajectory

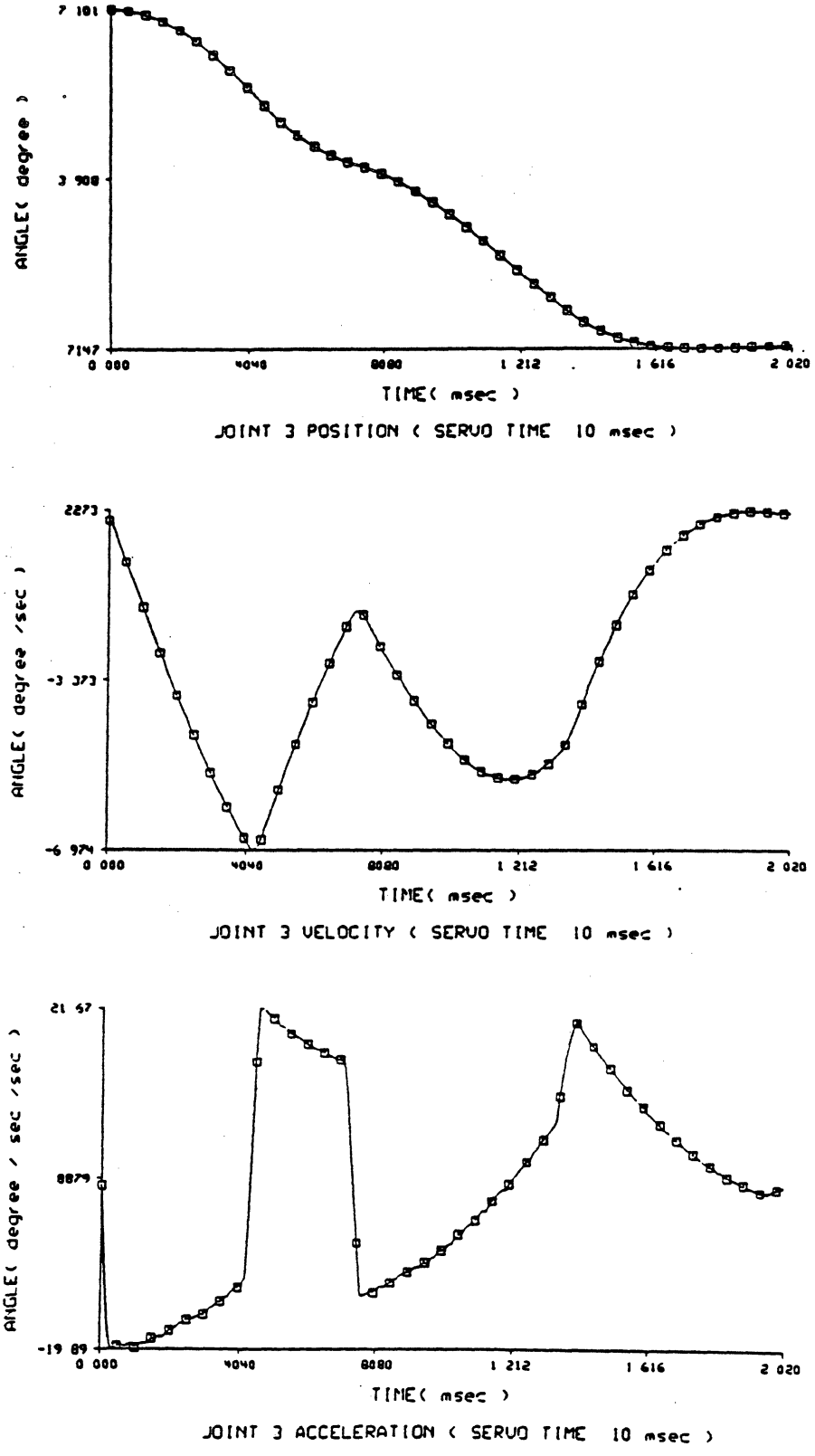


Figure 6.34 Time Scheduled Joint 3 Trajectory

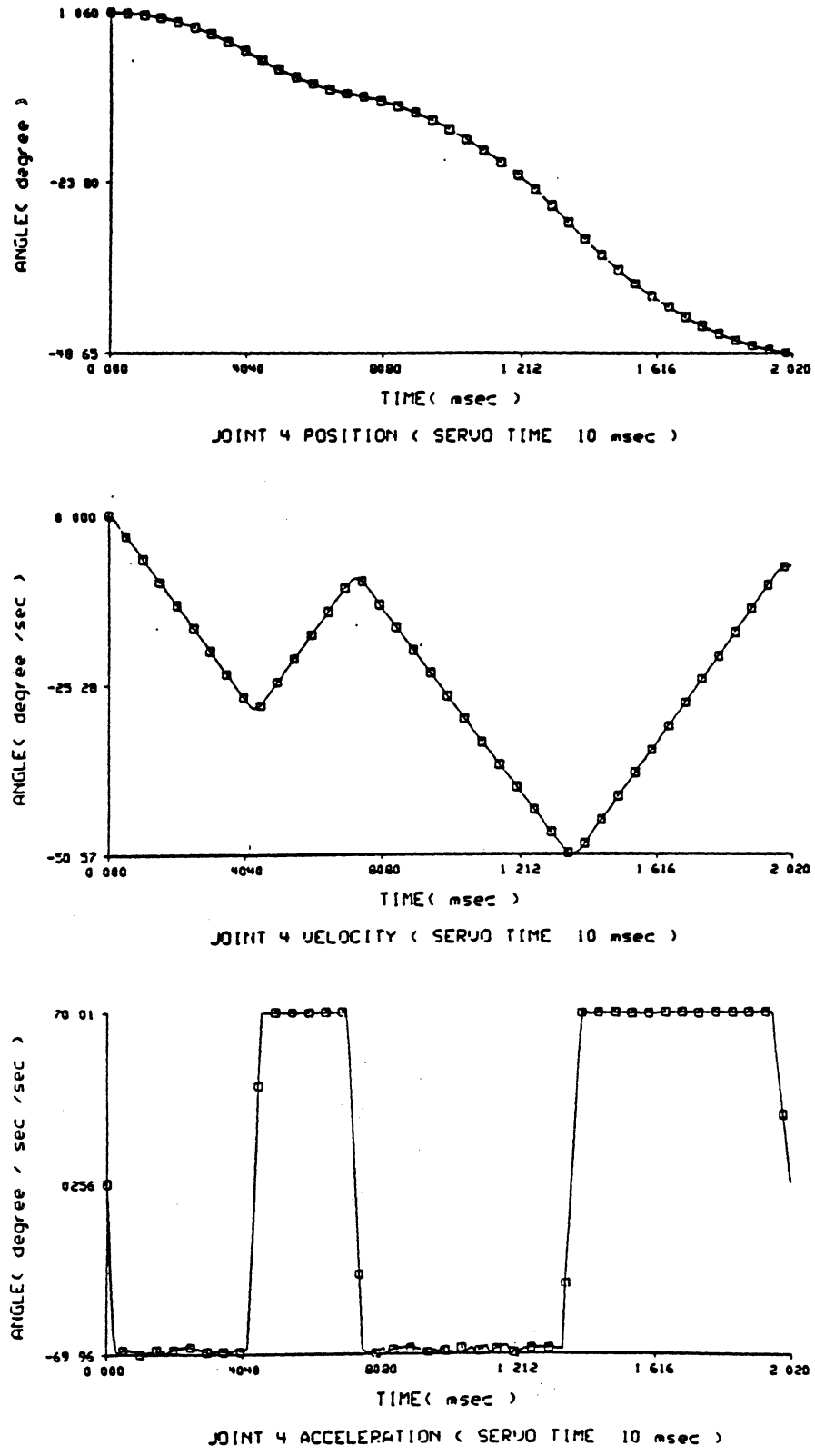


Figure 6.35 Time Scheduled Joint 4 Trajectory

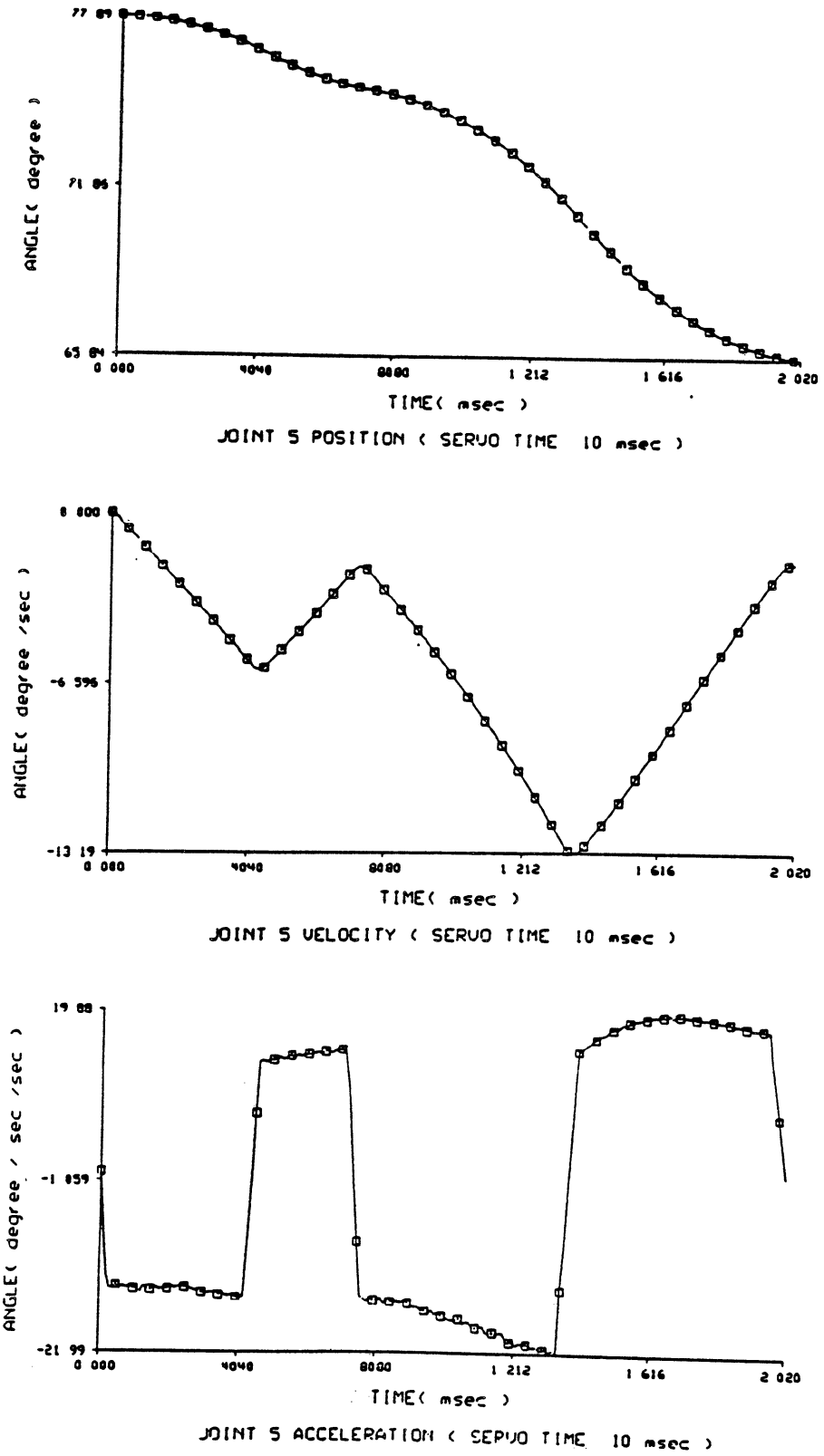


Figure 6.36 Time Scheduled Joint 5 Trajectory

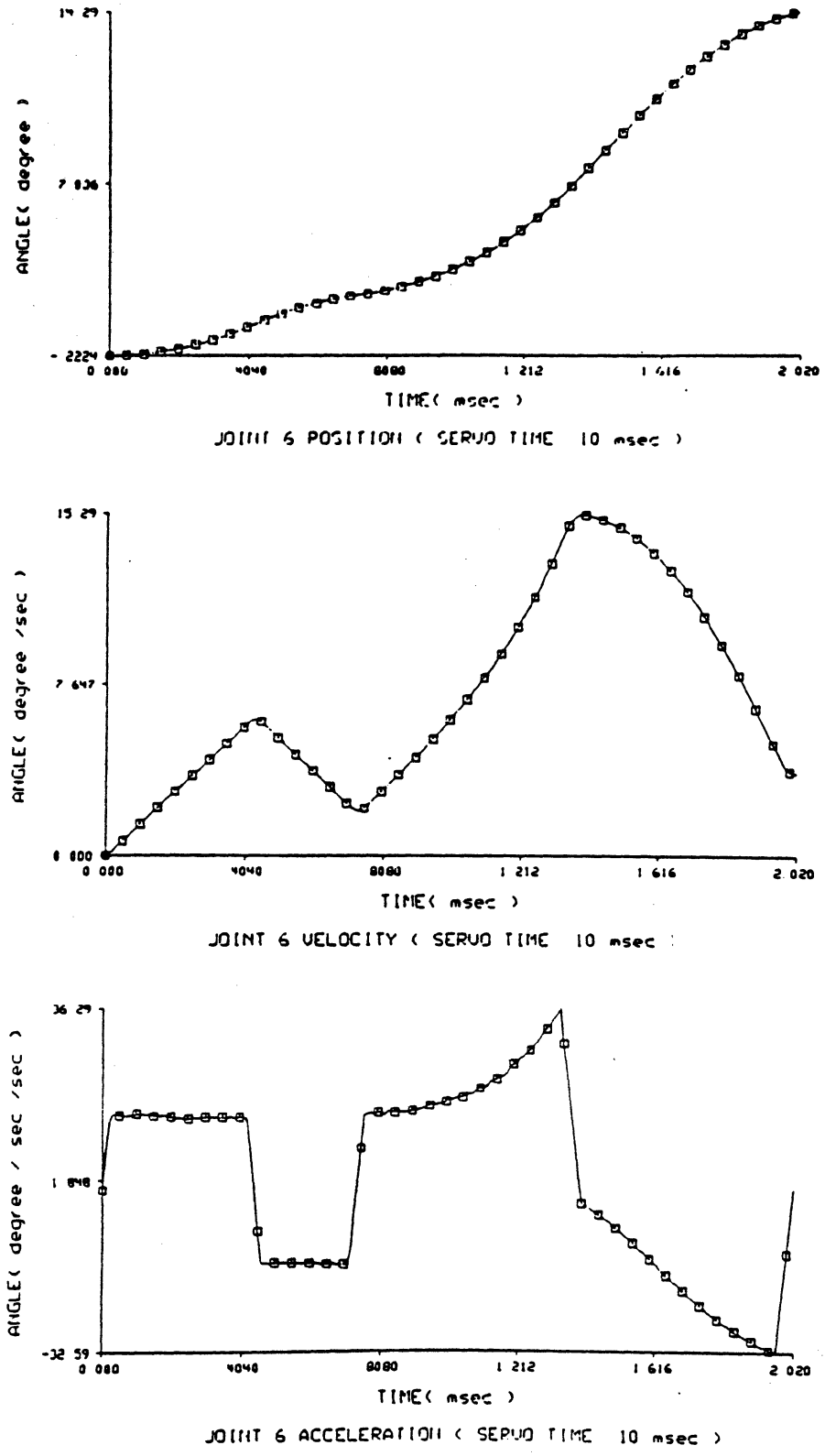


Figure 6.37 Time Scheduled Joint 6 Trajectory

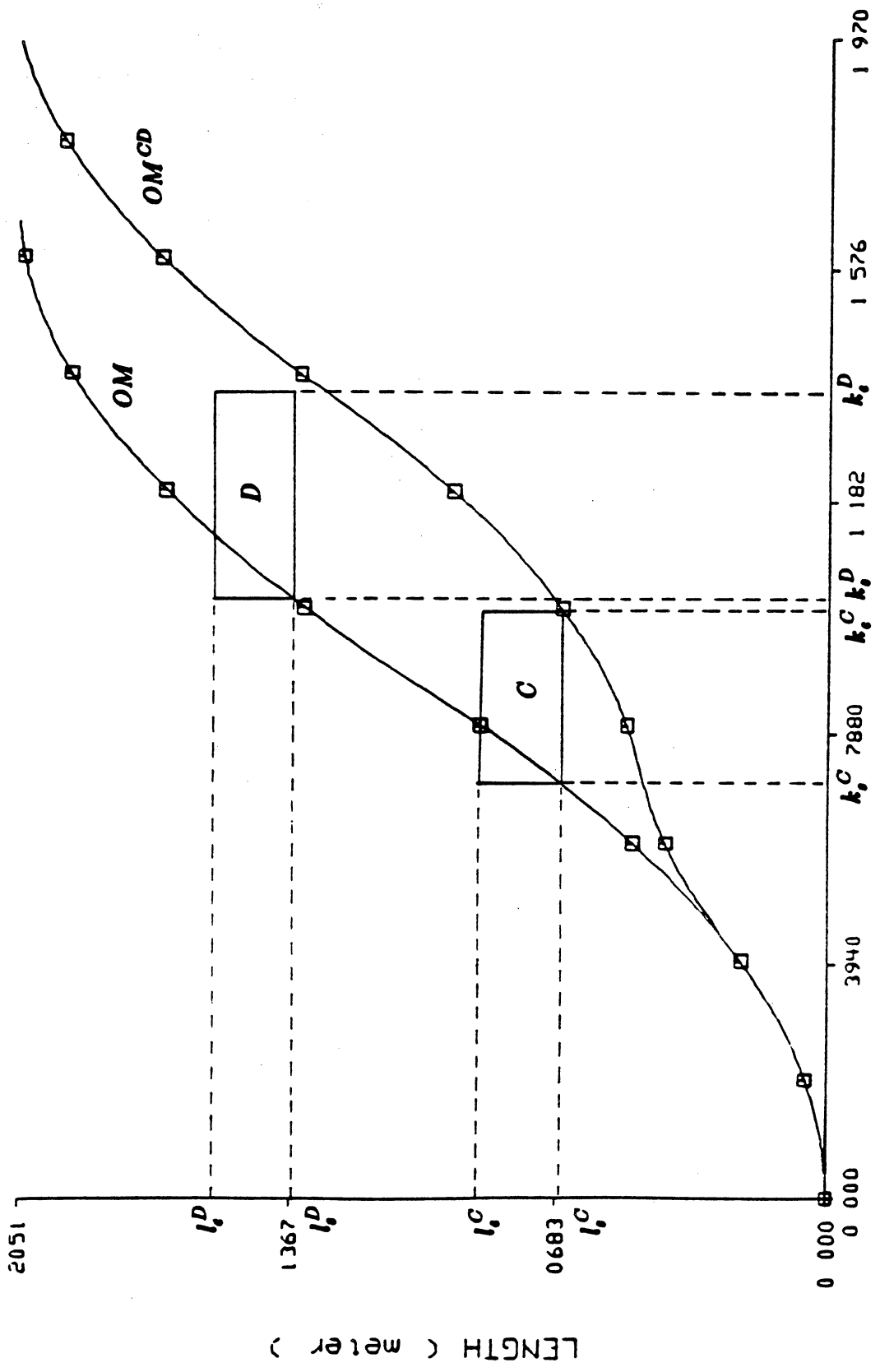


Figure 6.38 Comparison of Final Arrival Times

6.5. Summary

In this chapter, the proposed trajectory planning scheme of a straight line path was simulated on a VAX-11/780 computer. The error between the final location of the trajectory set point and the desired final location was found to be dependent on the location of the break point. Several servo time intervals were used in the simulation to investigate the effects of discrete time approximation of joint velocity, acceleration, and jerk. It was found that the servo time intervals, 10 *msec* and 20 *msec*, were successful in generating the control set points subject to the smoothness constraints while the 30 *msec* servo time interval leads to the breakdown of the search algorithms.

The existence of the straight line trajectory was investigated by using the procedure ESLT. Connected straight line segments were also simulated and the Cartesian deviation error at the intersection point was found to be less than 1 *mm*. The time scheduling of the trajectory was performed in an assumed collision situation. The traveled length versus servo time curve after time scheduling was found able to avoid the collision region in the collision map.

CHAPTER 7

DISCUSSION

7.1. Summary and Conclusion

In this thesis, we discussed an approach to the motion planning and motion control of two robots in a common workspace. The approach involves an investigation of straight line trajectory planning, collision-free path planning, and an adaptive control strategy in Cartesian space for tracking the collision-free trajectory faithfully under various payloads. Research results from these three areas constitute the major contribution of the thesis and they are briefly summarized below.

A discrete time trajectory planning scheme was developed for the straight line path. A real time servo interval instead of a normalized time interval is used for planning the joint trajectory. The planning of straight line trajectory is performed in Cartesian space and requires maximization of the local speed between two consecutive Cartesian set points. All the Cartesian set points are found on the straight line path subject to the torque and smoothness constraints.

An iterative forward and backward search algorithm was developed to determine the joint values at each servo interval. Break point and relaxation points are introduced to satisfy the boundary conditions at the final location of the straight line path. The resultant straight line trajectory is composed of three segments: an

acceleration portion, a deceleration portion, and a relaxation portion. The control set points on the acceleration and deceleration portions satisfy the straight line requirement while those on the relaxation portion do not. The existence of a straight line trajectory was also discussed and a procedure was developed to plan a straight line trajectory when the application of the search algorithms FW and MFW was not successful.

The sets of joint position, velocity and acceleration can be used as reference inputs to the manipulator controller in the joint-variable space. Also, the corresponding sets of Cartesian position, velocity, and acceleration can be used as reference inputs to the manipulator controller in the Cartesian space. The proposed straight line trajectory planning scheme was simulated on a VAX-11/780 computer to verify its performance.

Using the straight line trajectory and a sphere model for the manipulator wrist, potential collisions can be detected easily by checking the distance between the origins of the two spheres. For the collision avoidance of a moving robot in the presence of a stationary robot, a collision-free path was found subject to a performance measure. For the collision avoidance of two moving robots, a notion of a collision map is introduced which incorporates the location and the corresponding servo time information of the robots simultaneously.

Several collision situations have been identified when two robots are moving simultaneously. When the path modification was not allowed for collision avoidance, the collision map was utilized to plan a collision-free time scheduling of the trajectory on the same straight line path. When the path modification was allowed for collision avoidance, a procedure was developed to replan a collision-free

path subject to a performance measure and a constraint on time scheduling. The time scheduling of a straight line trajectory was performed on a VAX-11/780 computer to show its application.

For real time tracking of the desired collision-free trajectory, an adaptive perturbation control was developed in the Cartesian space. The control method adopts the ideas of resolved motion rate control and resolved motion acceleration control. Equations of motion of the manipulator were developed in Cartesian coordinates from which the linearized perturbation equations were derived along the desired hand trajectory.

The control system is characterized by feedforward and feedback components which can be computed separately and simultaneously. The feedforward component computes the nominal torques from the Newton-Euler equations of motion using the joint trajectory information from the trajectory planner. The feedback component, consisting of a recursive least square identification scheme and an optimal adaptive self-tuning controller for the linearized system, computes the perturbation torques which reduce the manipulator hand position and velocity errors along the nominal hand trajectory. A feasibility study of implementing the adaptive control for a six-joint PUMA manipulator was explored using present-day low-cost microprocessors.

7.2. Future Research

We discuss future research issues with possible extensions of this thesis. The future research issues are not difficult to think of. These include the following:

(a) Simulation of the proposed straight line trajectory planning scheme with actual torque constraints, which depends on the instantaneous joint position and

velocity.

(b) Investigation of a collision-free path for a moving robot in the presence of a stationary robot subject to different performance measures.

(c) Investigation of a collision-free path for two moving robots subject to different performance measures.

(d) Simulation and implementation of the proposed Cartesian space control method.

(e) Investigation of collision-free path and trajectory planning scheme in multiple robot system (robotic system with three or more robots).

(f) Investigation of other wrist modeling of various robots and investigation of the resultant computational complexity for the detection of potential collisions.

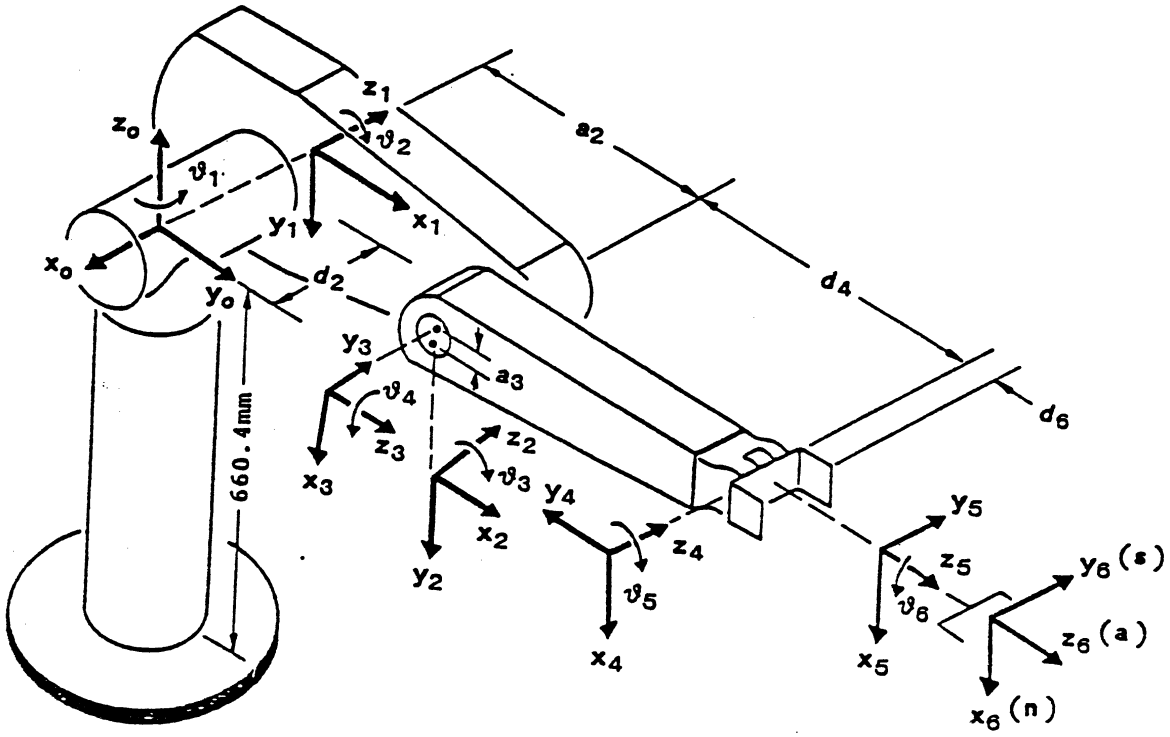
(g) Investigation of the time scheduling effects on the performance of multiple robot system.

(h) Investigation of a collision-free path of the two moving robots considering the whole arm of a manipulator.

APPENDICES

APPENDIX A

A PUMA Robot Arm Configuration



PUMA Robot Arm Link Coordinate Parameters

Joint i	ϑ_i	α_i	a_i	d_i	Range
1	90	-90	0	0	-160 to +160
2	0	0	432 mm	149.5 mm	-225 to +45
3	90	90	- 20.32 mm	0	-45 to +225
4	0	-90	0	432.0	-110 to +170
5	0	90	0	0	-100 to +100
6	0	0	0	56.5	-266 to +266

APPENDIX B

Homogeneous Transformation Matrix of a PUMA Robot

From equation (2.1), the 4×4 hand transformation matrix $\mathbf{H}(t)$ for the PUMA robot in Appendix A can be written as the following:

$$\mathbf{H}(t) = \begin{bmatrix} n_x(t) & s_x(t) & a_x(t) & p_x(t) \\ n_y(t) & s_y(t) & a_y(t) & p_y(t) \\ n_z(t) & s_z(t) & a_z(t) & p_z(t) \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

where

$$n_x(t) = C_1[C_{23}(C_4C_5C_6 - S_rS_6) - S_{23}S_5C_6] - S_1(S_4C_5C_6 + C_4S_6)$$

$$n_y(t) = S_1[C_{23}(C_4C_5C_6 - S_rS_6) - S_{23}S_5C_6] + C_1(S_4C_5C_6 + C_4S_6)$$

$$n_z(t) = -S_{23}(C_4C_5C_6 - S_4S_6) - C_{23}S_5C_6$$

$$s_x(t) = C_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] - S_1(-S_4C_5S_6 + C_4C_6)$$

$$s_y(t) = S_1[-C_{23}(C_4C_5S_6 + S_4C_6) + S_{23}S_5S_6] + C_1(-S_4C_5S_6 + C_4C_6)$$

$$s_z(t) = S_{23}(C_4C_5S_6 + S_4C_6) + C_{23}S_5S_6$$

$$a_x(t) = C_1(C_{23}C_4S_5 + S_{23}C_5) - S_1S_4S_5$$

$$a_y(t) = S_1(C_{23}C_4S_5 + S_{23}C_5) + C_1S_4S_5$$

$$a_z(t) = -S_{23}C_4S_5 + C_{23}C_5$$

$$p_x(t) = C_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] - S_1(d_6S_4S_5 + d_2)$$

$$p_y(t) = S_1[d_6(C_{23}C_4S_5 + S_{23}C_5) + S_{23}d_4 + a_3C_{23} + a_2C_2] + C_1(d_6S_4S_5 + d_2)$$

$$p_z(t) = d_6(C_{23}C_5 - S_{23}C_4S_5) + C_{23}d_4 - a_3S_{23} - a_2S_2$$

where $C_i \equiv \cos\theta_i$; $S_i \equiv \sin\theta_i$; $C_{ij} \equiv \cos(\theta_i + \theta_j)$; $S_{ij} \equiv \sin(\theta_i + \theta_j)$

APPENDIX C

Kinematic Equations for Euler Angles

Using the elements of the homogeneous transformation matrix in Appendix B, we can obtain:

$$\alpha(t) = \tan^{-1}\left[\frac{a_x S\gamma + a_y C\gamma}{-s_x S\gamma + s_y C\gamma}\right]$$

$$\beta(t) = \tan^{-1}\left[\frac{-n_z}{n_x C\gamma + n_y S\gamma}\right]$$

$$\gamma(t) = \tan^{-1}\left[\frac{n_y}{n_x}\right]$$

APPENDIX D

Inverse Kinematic Equations for a PUMA Robot

Given the Euler Angles (α, β, γ), we have the following equations from equation (2.2):

$$n_x(t) = C\gamma C\beta$$
$$n_y(t) = S\gamma C\beta$$
$$n_z(t) = -S\beta$$
$$o_x(t) = -S\gamma C\alpha + C\gamma S\beta S\alpha$$

$$where \quad -\pi \leq \theta_1 \leq \pi$$

Joint 2 Solution

From the geometry of the first three links, we have

$$R = \sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}$$

$$\sin u = -\frac{p_z}{R} = -\frac{p_z}{\sqrt{p_x^2 + p_y^2 + p_z^2 - d_2^2}}$$

$$\cos u = -\frac{ARM \cdot \sqrt{p_x^2 + p_y^2 - d_2^2}}{\sqrt{p_x^2 + p_y^2 + p_z}}$$

$$\cos v = \frac{a_2^2 + R^2 - (d_4^2 +}{2a_2R}$$

$$\sin v = \sqrt{1 - \cos^2 v}$$

Then,

$$\sin\theta_2 = \sin u \cos v + (ARM \cdot ELBOW) \cos u \sin v$$

$$\cos\theta_2 = \cos u \cos v - (ARM \cdot ELBOW) \sin u \sin v$$

and

$$\theta_2 = \tan^{-1}\left[\frac{\sin\theta_2}{\cos\theta_2}\right]$$

Joint 3 Solution

Again, from the geometry of the first three links, we have

$$\cos w = \frac{a_w^2 + (d_4^2 + a_3^2) - R^2}{2a_2\sqrt{d_4^2 + a_3^2}}$$

$$\sin w = ARM \cdot ELBOW \cdot \sqrt{1 - \cos^2 w}$$

$$\sin x = \frac{d_4}{\sqrt{d_4^2 + a_3^2}}$$

$$\cos x = \frac{|a_3|}{\sqrt{d_4^2 + a_3^2}}$$

Then,

$$\sin\theta_3 = \sin w \cos x - \cos w \sin x$$
$$\cos\theta_3 = \cos w \cos x + \sin w \sin x$$

and

$$\theta_3 = \tan^{-1}\left[\frac{\sin\theta_3}{\cos\theta_3}\right]$$

Joint 4 Solution

$$\theta_4 = \tan^{-1}\left[\frac{C_1 a_y - S_1 a_x}{C_{23}(C_1 a_x + S_1 a_y) - S_{23} a_z}\right]$$

Joint 5 Solution

$$\theta_5 = \tan^{-1}\left[\frac{(C_1 C_{23} C_4 - S_1 S_4)a_x + (S_1 C_{23} C_4 + C_1 C_4)a_y - S_{23} C_4 a_z}{(C_1 a_x + S_1 a_y)S_{23} + C_{23} a_z}\right]$$

Joint 6 Solution

$$\theta_6 = \tan^{-1}\left[\frac{(-S_1 C_4 - C_1 C_{23} S_4)n_x + (C_1 C_4 - S_1 C_{23} S_4)n_y + S_{23} S_4 n_z}{(-S_1 C_4 - C_1 C_{23} S_4)s_x + (C_1 C_4 - S_1 C_{23} S_4)s_y + S_{23} S_4 s_z}\right]$$

APPENDIX E

Direction Number of a Line

For the definition of the direction number of a line, we define the direction cosines of the line first. The direction cosines of a line are defined as the cosines of its direction angles which range between 0 and π from the positive directions of the global coordinate frame axes, x_0, y_0, and z_0 to the positive direction of a given directed line in space. Then, the direction number of a line is defined as any triple of numbers except [0, 0, 0] which is proportional to the direction cosines of the line.

If we denote a plane equation as,

$$a_1 x + b_1 y + c_1 z = d_1$$

then [a_1, b_1, c_1] will be the direction number of any lines which are perpendicular to the plane. Let us denote two plane equations as,

$$a_1 x + b_1 y + c_1 z = d_1$$

$$a_2 x + b_2 y + c_2 z = d_2$$

Then, the direction number of the intersection line of two planes must be perpendicular to [a_1, b_1, c_1] and [a_2, b_2, c_2]. If we use [α_e, β_e, γ_e] to stand for the direction number of the intersection line, they must satisfy the following equations.

$$a_1 \alpha_e + b_1 \beta_e + c_1 \gamma_e = 0$$

$$a_2 \alpha_e + b_2 \beta_e + c_2 \gamma_e = 0$$

Therefore, the direction number [α_e, β_e, γ_e] must be a proportional set of

$$\left[\begin{vmatrix} b_1 c_1 \\ b_2 c_2 \end{vmatrix}, - \begin{vmatrix} a_1 c_1 \\ a_2 c_2 \end{vmatrix}, \begin{vmatrix} a_1 b_1 \\ a_2 b_2 \end{vmatrix} \right]$$

which is equal to

$$\left[b_1 c_2 - c_1 b_2 , c_1 a_2 - a_1 c_2 , a_1 b_2 - b_1 a_2 \right]$$

BIBLIOGRAPHY

1. N. Ahuja et al., "Interference Detection and Collision Avoidance Among Three Dimensinal Objects," *Proc. 1st Annual Nat. Conf. on Artificial Intell.*, August 1980, pp. 44-48.

2. A.S. Albus, "A New Approach to Manipulator Control (CMAC)," *Trans. of ASME, J. of Dyn. Sys., Meas. and Cont.*, Septmber 1975, pp. 220-227.

3. K.J. Astrom, "Theory and Application of Adaptive Control-A Survey," *Proc. of Int'l Federation of Automatic Control*, Great Britian, 1983, pp. 471-483.

4. A.J. Barbera, M.L. Fitzgerald and J.S. Albus, "Concept for a Real Time Sensory Interactive Control System Architecture," *Proc. of the 4th Southeastern Symp. on System Theory*, April 1982, pp. 121-126.

5. A. K. Bejczy, "Robot Arm Dynamics and Control," *Technical Memorandum 33-669, Jet Propulsion Laboratory*, February 1974.

6. Beyer, *Standard Math Tables*, CRC Press, 1978, pp. 283-316.

7. J.E. Boblow, S. Dubowsky and J.S. Gibson, "On the Optimal Control of Robotic Manipulators with Actuator Constraints," *Proc. of Automat. Contr. Conf.*, June 1983, pp. 782-787.

8. M. Brady, et al. (editors), *Motion Planning*, MIT Press, 1982, pp. 221-243.

9. R. A. Brooks, "Solving the Find-Path Problem by Good Representation of Free Space," *IEEE Trans. on Systems, Man, and Cybernetics*, vol. SMC-13, No. 3, March 1983. pp. 190-197.

10. R. A. Brooks, "Planning Collision-Free Motions for Pick-and-Place Operations," *International Journal of Robotics Research,* vol. 2., No. 4, Winter 1983, pp. 19-44.

11. R. T. Chien et. al., "Planning Collision-Free Paths for Robotic Arm among Obstacles," *IEEE Trans. on Pattern Analysis and Machine Intelligence,* vol. PAMI-6, No. 1, January 1984, pp. 91-96.

12. E. Delp, "The CRC Plotting Package," *Computing Research Lab., Univ. of Michigan,* CRL-TR-14-83, 1983

13. S. Dubowsky, D. T. DesForges, "The Application of Model Referenced Adaptive Control to Robotic Manipulators," *Transaction of the ASME, Journal of Dynamic Systems, Measurement and Control,* Vol. 101, September 1979, pp 193-200.

14. P. Eykhoff, *System Identification: Parameter and State Estimation,* Wiley-Interscience, 1974, pp 240-241.

15. R. A. Frazer et al., *Elementary Matrices,* Cambridge University Press, 1960, pp 250-256.

16. E. Freund, "Nonlinear Control with Hierarchy for Coordinated Operation of Robots," *NATO Advanced Studies Institute on Robotics and Artificial Intelligence,* June 26-July 8, 1983, Italy.

17. E. Freund, "Hierarchical Control of Guided Collision Avoidance for Robots in Assembly Automation," *Proc. of 4th Int'l Conf. on Assembly Automation," pp. 91-103, September 1983. Japan*

18. E.G. Gilbert and I.J. Ha, "An Approach to Nonlinear Feedback Control with Applications to Robotics," *CRIM, Robot System Division RSD-TR-4-83,* Univ. of Michigan, Ann Arbor, 1983, and *IEEE Trans. on Syst. Man and Cyber.* vol. SMC-14, No. 6, Nov/Dec 1984, pp 879-884.

19. E.G. Gilbert and D.W. Johnson, "Distance Functions and Their Application to Robot Path Planning in the presence of Obstacles," *Proc. of 18th Annual Conf. on Information Sciences and Systems,* Princeton Univ., March 1984

20. J. M. Hollerbach, "Dynamic Scaling of Manipulator Trajectories," *Trans. of ASME, Journal of Dynamic Systems, Measurement, and Control* vol. 106, March 1984, pp 102-106.

21. R. Horowitz and M. Tomizuka, "An Adaptive Control Scheme for Mechanical Manipulators," *Trans. of ASME, Winter Meeting,* November 1980, Chicago

22. M. E. Kahn, B. Roth, "The Near-Minimum-Time Control of Open-Loop Articulated Kinematic Chains," *Transaction of the ASME, Journal of Dynamic Systems, Measurement and Control,* Vol. 93, September 1971, pp 164-172.

23. A. J. Koivo, T. H. Guo, "Adaptive Linear Controller for Robotic Manipulators," *IEEE Transactions on Automatic Control,* Vol. AC-28, No. 1, February 1983, pp 162-171.

24. C. S. G. Lee, "Robot Arm Kinematics, Dynamics, and Control", *IEEE Computer,* Vol. 15, No. 12, December 1982, pp 62-80.

25. C. S. G. Lee and M. J. Chung, "An Adaptive Control Strategy for Mechanical Manipulators," *IEEE Trans. Auto. Contr.,* Vol. AC-29, No. 9, pp. 837-840,

September 1984.

26. C. S. G. Lee, M. J. Chung, and B. H. Lee, "An Approach of Adaptive Control for Robot Manipulators," *Journal of Robotic Systems*, Spring 1984, pp. 27-57.

27. C. S. G. Lee, K. S. Fu, and R. C. Gonzalez, "Tutorials in Robotics," *IEEE Computer Press*, November 1983.

28. C. S. G. Lee and B. H. Lee, "Resolved Motion Adaptive Control for Mechanical Manipulators," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, Vol. 106, No. 2, pp. 134-142, June 1984.

29. C. S. G. Lee and B. H. Lee, "Planning of Straight Line Manipulator Trajectory with Torque Constraints," *Proceedings of the 23rd CDC Conference*, Las Vegas, December 1984.

30. C. S. G. Lee, B. H. Lee, and R. Nigam, "Development of Generalized D'Alembert Equations of Motion for Mechanical Manipulators," *Proceedings of the 22nd CDC Conference*, San Antonio, Texas, December 14-16, 1983.

31. C. S. G. Lee, T. N. Mudge, J. L. Turney, "Hierarchical Control Structure using Special Purpose Processors for the Control of Robot Arms," *Proceedings of the 1982 Pattern Recognition and Image Processing Conference*, Las Vegas, Nevada, June 14-17, 1982, pp 634-640.

32. C.S.G. Lee and M. Ziegler, "A Geometric Approach in Solving the Inverse Kinematics of PUMA Robots," *IEEE Trans. on Aerospace and Electronic Systems*, vol. AES-20, No. 6, November 1984, pp. 695-706.

33. R.A. Lewis, "Autonomous Manipulation on a Robot," *Tech. Momo of JPL,* California Institute of Technology, Memo-33-679, March 1974.

34. C. S. Lin, P. R. Chang, and J. Y. S. Luh "Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots," *IEEE Trans. Auto. Contr.,* Vol. AC-28, No. 12, December 1983, pp. 1066-1073.

35. T. Lozano-Peres, "Spatial Planning: A Configuration Space Approach," *IEEE Trans. on Computers,* vol. C-32, No. 2, February 1983, pp. 108-120.

36. T. Lozano-Peres, "Automatic Planning of Manipulator Transfer Movements," *IEEE Trans. on Sys., Man, and Cybern.,* vol. SMC-11, pp. 681-698, October 1981.

37. J. Y. S. Luh, "An Anatomy of Industrial Robots and Their Controls," *IEEE Trans. on Auto. Control,* vol. 28, No. 2, February 1983, pp. 133-153.

38. J. Y. S. Luh, M.W.Walker, and R.P.C. Paul, "On-Line Computational Scheme for Mechanical Manipulators," *Trans. of ASME, J. of Dynamic Sys., Meas. and Cont.,* vol. 102, June 1980, pp.69-76.

39. J. Y. S. Luh and C. E. Campbell, "Collision-Free Path Planning for Industrial Robots," *Proc. of the 20th CDC,* 1982, pp. 84-88.

40. J. Y. S. Luh and C.E. Campbell, "Minimum Distance Collision-Free Path Planning for Industrial Robots with a Prismatic Joint," *IEEE Trans. on Automatic Control,* vol. AC-29, August 1984, pp. 675-680.

41. J. Y. S. Luh and C. S. Lin, "Optimum Path Planning for Mechanical Manipulators," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, Vol. 102, June 1981, pp. 142-151.

42. J. Y. S. Luh and C. S. Lin, "Approximate Joint Trajectories for Control of Industrial Robots along Cartesian Path," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-14, No. 3, May/June 1984, pp. 444-450.

43. J. Y. S. Luh, M. W. Walker, R. P. Paul, "Resolved-Acceleration Control of Mechanical Manipulators," *IEEE Transactions on Automatic Control*, Vol. AC-25, No. 3, June 1980, pp 468-474.

44. B. R. Markiewicz, "Analysis of the Computed Torque Drive Method and Comparison with Conventional Position Servo for a Computer-Controlled Manipulator," *Technical Memo*, Momo-33-601, Jet Propulsion Lab, March 1973.

45. D. Nitzan and C.A. Rosen, "Programmable Industrial Automation," *IEEE Trans. on Computer*, vol.25, No.12, December 1976, pp. 1259-1270.

46. R. P. Paul, *Robot Manipulators: Mathematics, Programming, and Control*, MIT Press, 1981.

47. R. P. Paul, "Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm," *Stanford Artificial Intelligence Laboratory*, A.I. Memo 177, September 1972.

48. R. P. Paul, "Manipulator Cartesian Path Control," *IEEE Trans. Syst., Man, Cybern.*, Vol. SMC-9, No. 11, November 1979, pp. 702-711.

49. Alan de Pennington et. al., "Geometric Modelling: A Contribution Towards Intelligent Robots,"
Proc. of the 13th ISIR/Robot 7 Conf., pp. 7.35-7.54.

50. A.A. Petrov and I.M. Sirota, "Obstacle Avoidance by a Robot Manipulator under Limited Information about the Environment," *Avtomatika i Telemekhanika*, No.4, pp. 29-40, Moscow, April 1983.

51. G.N. Saridis, "Intelligent Robotic Control," *IEEE Trans. on Automatic Control, vol. 28, No.5*, May 1983, pp. 547-557.

52. G. N. Saridis, R. N. Lobbia, "Parameter Identification and Control of Linear Discrete-Time Systems," *IEEE Transactions on Automatic Control*, Vol. AC-17, No.1 February 1972, pp 52-60.

53. K.G. Shin and N.D. Mckay, "Minimum-time Control of Robotic Manipulators with Geometric Path Constraints," *to appear in IEEE Trans. on Automatic Control.*

54. R.C. Smith and D. Nitzan, "A Moduler Programmable Assembly Station," *Proc. of 13th ISIR/Robot 7*, Chicago, IL, April 1983.

55. M. Shneier, et al., "Robot Sensing for a Hierarchical Control System," *Proc. of 13th ISIR/Robot 7*, Chicago, IL, April 1983.

56. J.A. Simpson, R.J. Hooken and J.S. Albus, "The Automated Manufacturing Research Facility of NBS," *Journal of Manufacturing System*, vol. 1, No. 1, 1982.

57. M. Takegaki, S. Arimoto, "A New Feedback Method for Dynamic Control of Manipulators," *Transactions of the ASME, Journal of Dynamic Systems, Measurement, and Control*, Vol. 102, June 1981, pp 119-125.

58. R. H. Taylor, "Planning and Execution of Straight Line Manipulator Trajectories," *IBM J. Res. Develop.*, Vol. 23, No. 4, July 1979, pp. 424-436.

59. Udupa, S. M. "Collision Detection and Avoidance in Computer Controlled Manipulators," *Proc. 5th Int. Joint Conf. Artificial Intell.*, Los Altos, California, 1977, pp. 737-748.

60. R. A. Volz, T. N. Mudge, and D. A. Gal, "Using Ada as a Programming Language for Robot-Based Manufacturing Cells," *IEEE Trans. on System, Man, and Cybernetics*, vol. SMC-14, No. 6, Nov/Dec 1984, pp 863-878.

61. D.E. Whitney, et al., "Design and Control of Adaptable Programmable Assembly Systems," *NSF 1st Report, R-1284, Charles Stark Draper Lab.*, august 1979.

62. D. E. Whitney, "Resolved Motion Rate Control of Manipulators and Human Prostheses," *IEEE Transactions on Man-Machine System*, Vol. MMS-10, No. 2, June 1969, pp 47-53.

63. D. E. Whitney, "The Mathematics of Coordinated Control of Prosthetic Arms and Manipulators," *Trans. ASME, J. Dynamic Syst., Meas., Contr.*, December 1972, pp. 303-309.

64. C. H. Wu, R. P. Paul, "Resolved Motion Force Control of Robotic Manipulator," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. SMC-12,

No. 3, June 1982, pp 266-275.

65. K. K. D. Young, "Controller Design for a Manipulator Using Theory of Variable Structure Systems," *IEEE Transactions on Systems, Man, and Cybernetics*, Vol. SMC-8, No. 2, February 1978, pp. 101-109.