AN ABSTRACT OF THE THESIS OF

Yuksel Yildirim for the degree of Master of Science in Department of Mechanical Engineering presented on February 10, 1989.

Title: Optimization of Polynomial Trajectories for Robotic Manipulators

Abstract approved: Redacted for Privacy

Dr. Chinyere Onwubiko

In this thesis, a method is presented to construct minimum-time robot trajectories for predefined Cartesian end-effector path in a workspace containing obstacles. The method is preferably applied to a geometric collision-free path of a SCARA robot by using the theories of Bezier, B-spline, and parabolic blending curves. The motion of the manipulator is defined by Cartesian control points and represented by a sequence of generated Cartesian knots along the end-effector path which is transformed into sets of joint displacements with one set for each joint. Cubic spline functions are then used to fit the sequence of joint displacements for each joint. The minimum-time trajectory planning problem is formulated as the problem of minimizing the total traveling time subject to constraints on joint positions, velocities, accelerations, jerks, torques and end-effector acceleration. The modified pattern search goal programming method is proposed to solve this nonlinear optimization problem. The computer program, ROBOPATH, has been written to implement the algorithm for a manipulator with two

links and two degrees of freedom. The examples show the algorithm to be a useful tool in the design of manipulators, robot tasks and workcells. The results show that different locations of a path and different path shapes resulted in different total traveling times. Also, as a result, approximation curve techniques gave shorter total traveling time than the interpolation curve technique.

Optimization of Polynomial Trajectories for Robotic Manipulators

by

Yuksel Yildirim

A THESIS

submitted to

Oregon State University

in partial fulfillment of
the requirements for the
degree of

Master of Science

Completed February 10, 1989

Commencement June 1989

APPROVED:

Redacted for Privacy

Professor of Department of Mechanical Engineering in charge of major

Redacted for Privacy

Head of Department of Mechanical Engineering

Redacted for Privacy

Dean of Graduate School

Date thesis is presented February 10, 1989

Typed by Cindy Withrow for Yuksel Yildirim

This thesis is dedicated to my parents who made many sacrifices in order that I could pursue my studies.

ACKNOWLEDGEMENT

The theory of minimum-time trajectory planning and their applications is a relatively recent development. As late as 1980, there were no more than a handful of papers mentioning robot path planning and trajectory planning by name. Today, less than eight years later, there are well over 100 research papers on the subject, and it still remains an active research area.

The rapid development of trajectory and path plannings is due primarily to their great usefulness in robot applications. It appears that the most turbulent years in the development of trajectory and path plannings are over. It is now generally agreed that they will become the firmly entrenched tools in the design of robotic manipulators and their workcells. Thus, my aim here is to present a fairly complete and unified treatment of minimum-time trajectory planning for robotic manipulators, which, I hope, will prove to be a useful source of information for engineers and scientists.

I would like to take this opportunity to acknowledge some of the individuals and institutions that have been of assistance in the preparation of this thesis. First, I would like to thank Professor Eugene F. Fichter for introducing me to robotics and polynomial curve techniques, and Professor Chinyere Onwubiko for introducing me to computational geometry and optimization techniques. I also appreciate Professors Eugene F. Fichter and Chinyere Onwubiko for their helpful guidance and support while working on this research. I am grateful to Professor C. E. Smith for his useful discussions and suggestions. Finally, I would like to express my appreciation to Department of

Mechanical Engineering at Oregon State University for support of my research over the past one year.

I would especially like to mention my thanks to the fellow students in office R236 and R238 for their useful discussions, comments and suggestions, and for the long busy hours we have spent together. Finally, of course, I lovingly acknowledge the help of Yolanda Harris and Alp Malazgirt for their patience in the face of my emotional ups-and-downs throughout the entire process.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

OPTIMIZATION OF POLYNOMIAL TRAJECTORIES FOR ROBOTIC MANIPULATORS

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW

Robotic manipulators are employed in a wide range of industrial applications to perform jobs regarded as tedious for human operators. Robots are ideally suited for jobs which are unsafe or unhealthy for people. Non-industrial robot systems are used in space and undersea technologies. A typical robot workstation is shown in Figure 1.1 (Maus and Allsup 1986).

One important goal of today's robotic manipulators is to increase productivity in industrial and non-industrial applications. The speed and productivity of present robotic manipulators are restricted by the capabilities of their actuators. The capabilities of their actuators are usually functions of their dynamic state because robotic manipulators have nonlinear dynamic characteristics. However, increasing the actuator size and power is not the best solution (Bobrow et al. 1983 and 1985). It can be self-defeating to use larger actuators because of their increased inertia, higher cost and greater power consumption. Another way of attaining speed increase is to move robots in minimum time from their initial positions to their goal positions. In other words, to increase productivity we would like to increase the speed of robotic manipulators. In a structured workplace, the deviation from the desired path is undesirable and dangerous.

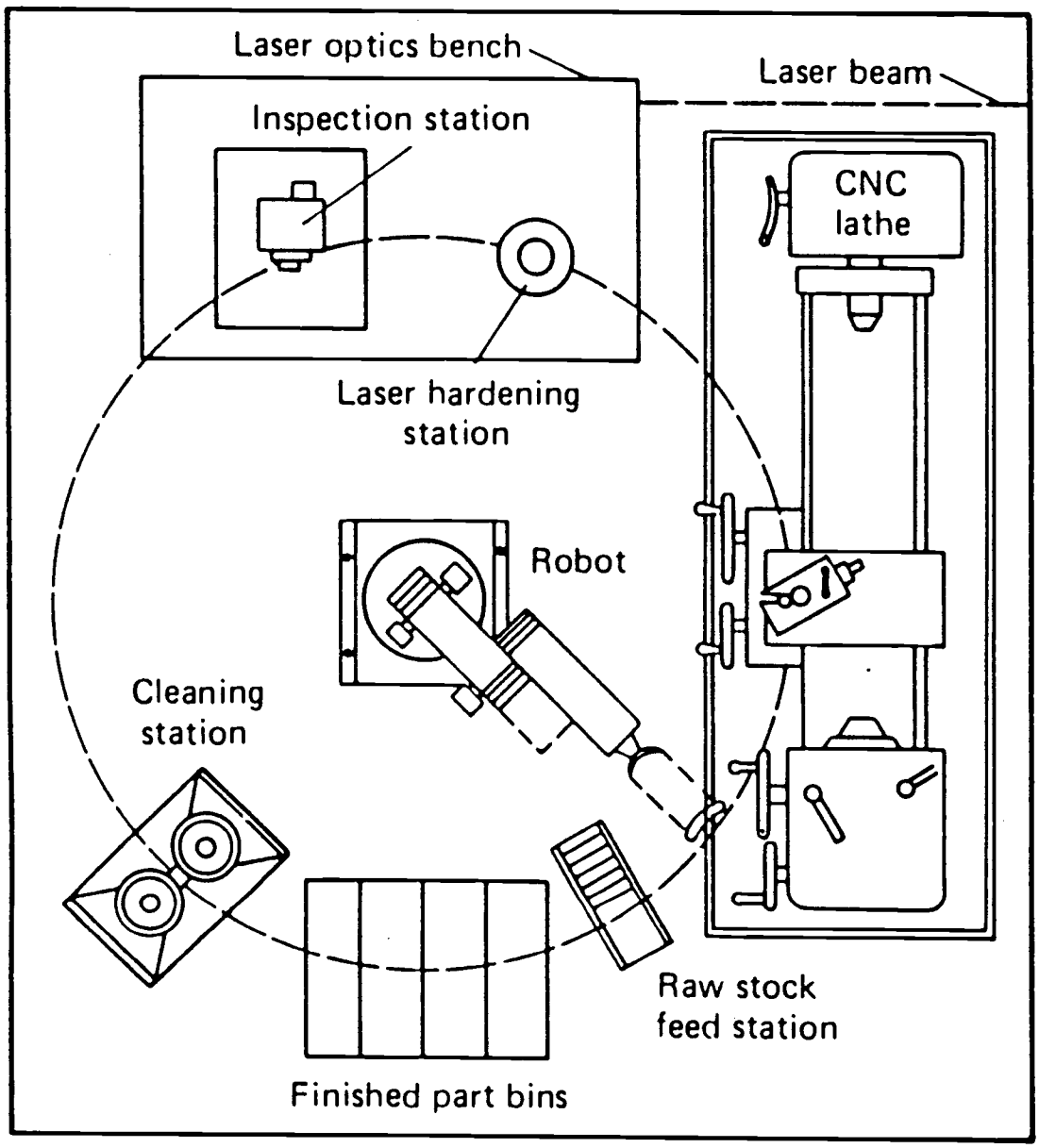


Figure 1.1: A robotic manipulator and its workstation.

The theory of minimum-time robot path planning and trajectory generation have recently received attention despite its importance. As late as 1980, there were only a few published papers mentioning robot path planning and trajectory generation by name. Today, less than nine years later, a lot of work has been done in path planning and trajectory generation. There are well over 100 research papers on the subject, and it still remains an active research area. On the other hand, a lot of work has been also done in the area of path control and tracking. A balanced combination of robot path control and path planning is necessary for both industrial and non-industrial applications with robotic manipulators. Therefore, the minimum time path planning should be aimed at this objective.

Trajectory planning is an important off-line stage which is concerned with the generation of a time history of a robot's joint position, velocity, acceleration, jerk, and input torques. Robot trajectory defines the time sequence of intermediate configurations in a desired motion. An executable trajectory should require the robot arm to move to a point inside its workspace or move with a velocity, acceleration or joint torque that is physically possible. The trajectory optimization is to find a trajectory function which satisfies the given constraints and minimizes the measure of the total traveling time.

It is necessary to specify the motion in much more detail than simply stating the desired final configuration. In order to do that, we must give a sequence of desired via points or intermediate points between the initial and final positions. Each of the path points is usually specified in terms of a desired position and orientation of the

end-effector frame relative to the base frame in Cartesian coordinates to easily visualize the correct end-effector configurations. If the joint coordinates are desired at the via points along the end-effector path, then each of the via points is converted into a set of desired joint angles using the inverse robot arm kinematics because it is generally difficult to convert a curve in Cartesian coordinates to that in joint coordinates. For trajectory planning in joint space, the time history of the manipulator end-effector's path is planned, and the corresponding joint positions, velocities, accelerations, and jerks are derived from the manipulator end-effector's path. Since controlled jerk may extend the life span of the robot, the joint jerks (the rate of change of joint accelerations) should be kept small to prevent mechanical parts from excessive wear and tear.

Prior to 1980, the minimum-time path for robotic manipulators was a long-standing and unsolved problem of considerable interest. The recent works by Luh and Lin (1981 and 1984), Lin, Chang, and Luh (1983), Lin and Chang (1985), Thompson and Patel (1985), and Braibant and Geradin (1987) have shown that an efficient approach to optimum control of manipulators may consist of splitting the problem into two tasks: off-line programming of a robot path along which an optimization can be achieved, followed by an on-line process during which the manipulator is assumed to track the path. The problem of tracking the pre-planned path is a servo-control problem which has been discussed by several authors and will not be considered here. In this study we are interested in developing suitable methods for defining and describing the desired motions of the end-effector or tool frame relative to the base frame between the path endpoints and wish to move the end-effector frame from

its initial position to a desired goal position in a smooth manner in a minimum time. In general, for most robotic manipulators, this motion basically involves a change in orientation as well as a change in position of the tool relative to the base. Also, we want the motion of the manipulator to be a smooth function which is continuous and has continuous first, and second derivatives. The third derivative may be also desired to be continuous. In order to guarantee smooth paths, we must put some sort of constraints on the spatial and temporal qualities of the path between the via points. In this study, the minimum-time trajectory planning is done in the joint space. We consider the problem of moving the manipulator end-effector from its initial position to a goal position as a motion of the end-effector frame relative to the base frame in minimum-time along a specified geometric path in Cartesian space. Enough Cartesian path points on the robot path are generated by using polynomial curve techniques and transformed into joint displacements. Cubic spline functions are used for constructing joint trajectories for robotic manipulators. Then our proposed algorithm schedules the time intervals between each pair of adjacent knots so that the total traveling time will be minimized subject to the constraints on joints positions, velocities, accelerations, jerks, and torques and the constraint on the end effector or tool frame acceleration. In addition, our method takes into account the full details of the non-linear dynamics of manipulator.

In this study, we apply our algorithm to the first two degrees of freedom of a four degree-of-freedom SCARA robot. Since the position and orientation of the gripper throughout the motion are about the vertical axis, we consider only the horizontal link motion in an xy-plane.

Therefore, we may treat it as a two degree-of-freedom planar manipulator.

For constructing end-effector paths, we use both interpolating and approximation curve techniques, which are currently used in CAD as powerful interactive design tools. In our proposed method, parametric polynomial functions such as parabolic blending, Bezier, and B-Spline curves are used to design trajectories for the end-effectors of robotic manipulators in Cartesian space. After transforming the Cartesian path points into joint space, piecewise cubic spline functions are used. They are expressed as functions of time to interpolate corresponding joint knot points for designing the joint trajectories. The Modified Pattern Search method with goal programming is used to minimize the total traveling time subject to joint constraints and end-effector acceleration constraint.

1.2 BACKGROUND

A number of techniques have been developed for planning minimum-time trajectories of industrial robots (Luh and Walker 1977; Luh and Lin 1981 and 1984; Lin, Chang, and Luh 1983; Lin and Chang 1985; Kim and Shin 1984 and 1985; Shin and McKay 1983, 1985, 1986a, and 1986b; Jeon and Eslami 1986), which usually entail complex computations and algebraic manipulations. Some of these methods make fairly specific assumptions about the form of the joint torque/force constraints, thereby limiting their applicability. A technique was developed to minimize the time required to move the robotic manipulator along a specified path consisting of straight lines and circular arcs by Luh and Walker (1977), and Luh and Lin (1981). In their work, piecewise

constant acceleration and maximum velocity constraints are assumed. Lin, Chang, and Luh (1983) described the construction of cubic spline joint trajectories with optimal placement of the knots. The cubic splines are used for constructing joint trajectories. Optimization is done by minimizing the total traveling time subject to constant limits on speed, acceleration, and jerk for each joint. This is a conservative estimate since the maximum acceleration of each joint depends on the configuration. Kim and Shin (1984 and 1985) developed a minimum-time path planning method in joint coordinates with the consideration of the manipulator dynamics as well as other realistic constraints which are those on the generated torques/forces and angular velocities and on the deviation at each corner point in the joint paths.

Shin and McKay (1984 and 1986a) proposed a robot path planning using dynamic programming to find the positions, velocities, accelerations, and torques that minimize cost of moving a robotic manipulator along a specified geometric path subject to input torque/force constraints. Also, Shin and McKay (1983 and 1985) presented a method for obtaining trajectories for minimum-time control of a mechanical arm given the desired geometric path and torque constraints. In addition, Shin and McKay (1986b and 1986c) developed a method for determining an approximate minimum-time geometric path for their previous trajectory planners in Shin and McKay (1983 and 1985) and presented a minimum-time trajectory planning scheme for actual implementation and general types of torque constraints. Jeon and Eslami (1986) proposed a method which minimizes the subtraveling and transition times subject to input torque constraints and yields an off-line minimum time joint trajectory planning. Paul (1979) proposed an on-line

Cartesian path tracking scheme. He described the design of manipulator Cartesian paths made up of straight-line segments for the hand motion. The velocity and acceleration of the hand between these segments are controlled by converting them into the joint coordinates and smoothed by a quadratic interpolation routine. Luh and Lin (1984) used the velocity, acceleration, and jerk bounds which are assumed constant for each joint. They selected several knot points on the desired Cartesian path, solved the inverse kinematics, and found appropriate smooth lower-degree polynomial functions to fit through these knot points in the joint coordinates.

Sahar and Hollerbach (1985) applied dynamic programming techniques to the nonlinear problem by searching a tessellated state space for the optimal trajectory. They presented that, in general, the optimal paths are not straight lines, but rather curves in joint space that utilize the dynamics of the manipulator and gravity to help in moving the arm faster to its destination. However, this method assumes straight line segments and thereby neglects the important effects of path curvature. Computation time for such an approach can be a problem if it is applied to realistic systems. Rajan (1985) presented an algorithm for determining the minimum time trajectory for a robot arm with actuator constraints by combining the control theory and the exhaustive search approaches. Lin and Chang (1985) developed a method for planning the point-to-point path that the robot manipulator may adopt to minimize a specified performance index, under the realistic physical constraints on joint torques, accelerations, velocities, and position limits. Cubic spline functions are proposed for an approximation. Thompson and Patel (1985) described a procedure for constructing robot joint trajectories

using the theory of B-splines. Their work is very similar to that of Lin, Chang, and Luh (1983). Chand and Doty (1985) described the construction of on-line cubic spline joint trajectories for a limited-point look ahead on a specified end-effector trajectory. Mizugaki et. al. (1985) presented the relationship between the shape of the path in Cartesian coordinates and the velocity distribution along the path. They proposed a method that makes a free curve trajectory using a Bezier curve by passing specified points and satisfying constraints on trajectory.

Paul and Zhang (1985) presented a method for the specification of the trajectories in a dynamics free manner in both fixed and moving coordinate systems. Seeger and Paul (1985) designed a control scheme to make a robot manipulator move as fast as possible within its velocity and acceleration constraints.

Recently, an optimal control algorithm for the minimum time trajectory along a specified path has been derived for the given actuator constraints. Bobrow (1982) and Bobrow, Dubowsky, and Gibson (1983 and 1985) presented an algorithm for computing the actuator torques that move the robotic manipulator along a specified path in minimum time, subject to constraints on the torques. Later, this algorithm was extended to include limits on the gripping force and the acceleration of the payload in addition to the actuator constraints by Shiller (1984), and Shiller and Dubowsky (1985). Their algorithm finds the minimum-time motions for a robotic manipulator between given end states. Dubowsky, Norris, and Shiller (1986) extended their previous method (Shiller 1984; Shiller and Dubowsky 1985), which also considers the full non-linear dynamics of the manipulator and the saturation

characteristics of its actuators, and finds the minimum-time motions for a robotic manipulator between given end states using a Bezier curve.

1.3 SCOPE OF THIS STUDY

The objective of this study is to design minimum-time polynomial trajectories for robotic manipulators. The study is basically done for a four degree-of-freedom SCARA robot.

The minimum-time trajectory planning algorithm is simulated on the first two degrees of freedom SCARA robot in a computer program, ROBOPATH. The inputs to the program include the manipulator's parameters, the control points of robot path in xy-plane, the constraints on joint positions, velocities, accelerations, jerks, and torques, the constraint on end-effector acceleration, and initial guess on time intervals for optimization of the trajectory. Then the program obtains the values of the minimum time intervals, the optimal joint positions, velocities, accelerations, jerks, and torques, end-effector accelerations along a specified polynomial path for the tip of the robotic manipulator or end-effector frame. The program is efficient and requiring less than five minutes of computation time on an IBM PC/AT microcomputer although computation time is not the key issue since the calculation is done off-line.

The implementation of this algorithm requires that the kinematics and dynamics of the robotic manipulator be generated in an explicit form. The derivation of the equations of motion for the first two degree-of-freedom of SCARA robot is quite easy. The equations of motion for SCARA robot are generated using the iterative Newton-Euler dynamic formulation. The Newton-Euler method to an n link open chain

manipulators is shown in Chapter 2. SCARA robot has revolute joints and symmetric links in its first two joints and links, respectively. The dimensions of the manipulator's links, linkage masses, and inertia properties are left variable.

In this study, interpolating and approximation curve techniques are used to design Cartesian end-effector paths. These curve techniques are parabolic blending, Bezier, and B-spline curves. Cartesian path points are generated by using one of these curve techniques. These Cartesian path points on the robot path are transformed into N sets of joint displacements, with one set for each of the N joints of the robot arm. Then piecewise cubic spline polynomials are used to interpolate the sequence of these corresponding joint displacements for each of the N joints to design the N joint trajectories of the robot arm. Spline functions are expressed in terms of time intervals between adjacent knots for joint trajectories. To minimize the total traveling time, the time intervals are adjusted subject to the joint constraints and end-effector acceleration constraint within each of the joint polynomial trajectory pieces. A large number of constraints are involved in the optimization process. In this study, the idea of Modified Pattern Search method with goal programming is applied for this problem by handling the constraints carefully.

In this work, the algorithm has been shown to be a useful tool in the design of manipulators and their workcells. It is also shown that different locations of a path and different path shapes of the end-effector along the path resulted in different total traveling times. Approximation curve techniques (Bezier and B-spline curves) gave shorter total traveling times than the interpolation curve technique (parabolic

blending curve). The final configuration of the manipulator may be chosen for best performance by iterating on the possible robot paths or robot arm designs.

CHAPTER 2

MANIPULATOR KINEMATICS AND DYNAMICS

2.1 INTRODUCTION

The central topic of this chapter is to provide a systematic methodology for the kinematic and dynamic analysis of manipulators. It is mainly divided into five sections: Introduction, Kinematics, Dynamics, Example of the 4-Link SCARA Robot, and Remarks.

In the kinematics section, two coordinate systems are used to describe the position of the manipulator: joint coordinates and link coordinates. The development of kinematics is based on the use of homogeneous transforms to describe the position and orientation of the link coordinates. Kinematics of manipulator deals with the analytical study of the geometry of motion of a manipulator with respect to a fixed reference coordinate system as a function of time without regard to the forces/moments which cause the manipulator motion. Direct kinematics of manipulator is concerned with computing the position and orientation of the manipulator's end-effector relative to the base of the manipulator as a function of the joint variables. Inverse kinematics of manipulator determines joint variables given a desired position and orientation of the end-effector of the manipulator and the geometric link parameters with respect to the base coordinates of the manipulator (Craig 1986; Fu et al. 1987).

The dynamics section presents the equations of motion of a manipulator which the optimization algorithm requires. In general, the dynamic performance of a manipulator directly depends on the efficiency of the control algorithms and dynamic model of the manipulator. The

actual dynamic model of a manipulator may be obtained using either the Lagrange or Newton-Euler formulations. It is shown how the equations in the kinematics section for position, velocity, and acceleration of the link coordinates can be used with the Newton-Euler equations of motion of an open-chain manipulator.

The method of Newton-Euler formulation is independent of the type of manipulator configuration. The Newton-Euler method involves the successive transformation of velocities and accelerations from the base of the manipulator out to the end-effector, link by link, using the relationships of moving coordinate systems. Forces are then transformed back from the end-effector to the base to obtain the joint torques (Luh, Walker, and Paul 1980; Brady et al. 1982; Craig 1986; Fu et al. 1987).

2.2 MANIPULATOR KINEMATICS

In this section, we consider position and orientation of the manipulator linkages in static situations. In order to deal with the complex geometry of a manipulator we will affix frames to the various parts of the mechanism and then describe the relationship between these frames (Craig 1986). The link frames are named by number according to the link to which they are attached. In other words, frame {i} is attached rigidly to link i. The homogeneous coordinates are used to represent position vectors in a three-dimensional space, and the rotation matrices are expanded to 4x4 homogeneous transformation matrices to include the translational operations of the body-attached coordinate frames. The advantage of using the Denavit-Hartenberg representation of linkages is its algorithmic universality in deriving the kinematic equation of a manipulator (Fu et al. 1987).

The convention described in Craig (1986) is used to define a frame attached to each link. It is briefly given in Appendix A. We strongly suggest reader to refer to the convention in Craig (1986). The manipulator Jacobian, which represents the relationship between the joint displacements and the end-effector location at the present position and the robot arm configuration, is given in Appendix B.

2.2.1 DIRECT KINEMATICS

Direct kinematics refers to the computation of the position or motion of each link as a function of the joint variables. In general, the transformation, which defines frame {i} relative to the frame {i-1}, is a function of four link parameters. Only one of these four parameters is variable for any given robot. If joint i is revolute joint, then θ_i is the joint variable and a_i, α_i, and d_i are constants. If joint i is prismatic, the d_i is the joint variable and a_i, α_i, and θ_i are constants.

Figure 2.1 (Craig 1986) shows the location of frames {i-1} and {i} for a general manipulator. If we want to write the transformation which transforms vectors defined in {i} to their description in {i-1}, we may write the ${}^{i-1}_{i}T$ matrix in the following form (Craig 1986):

$$ {}^{i-1}_{i}T = \mathrm{Rot}(\hat{X}_i,\alpha_{i-1})\mathrm{Trans}(\hat{X}_i,a_{i-1})\mathrm{Rot}(\hat{Z}_i,\theta_i)\mathrm{Trans}(\hat{Z}_i,d_i) \qquad (2.1) $$

We obtain the general form of ${}^{i-1}_{i}T$ in homogeneous matrix notation by multiplying out Equ. (2.1):

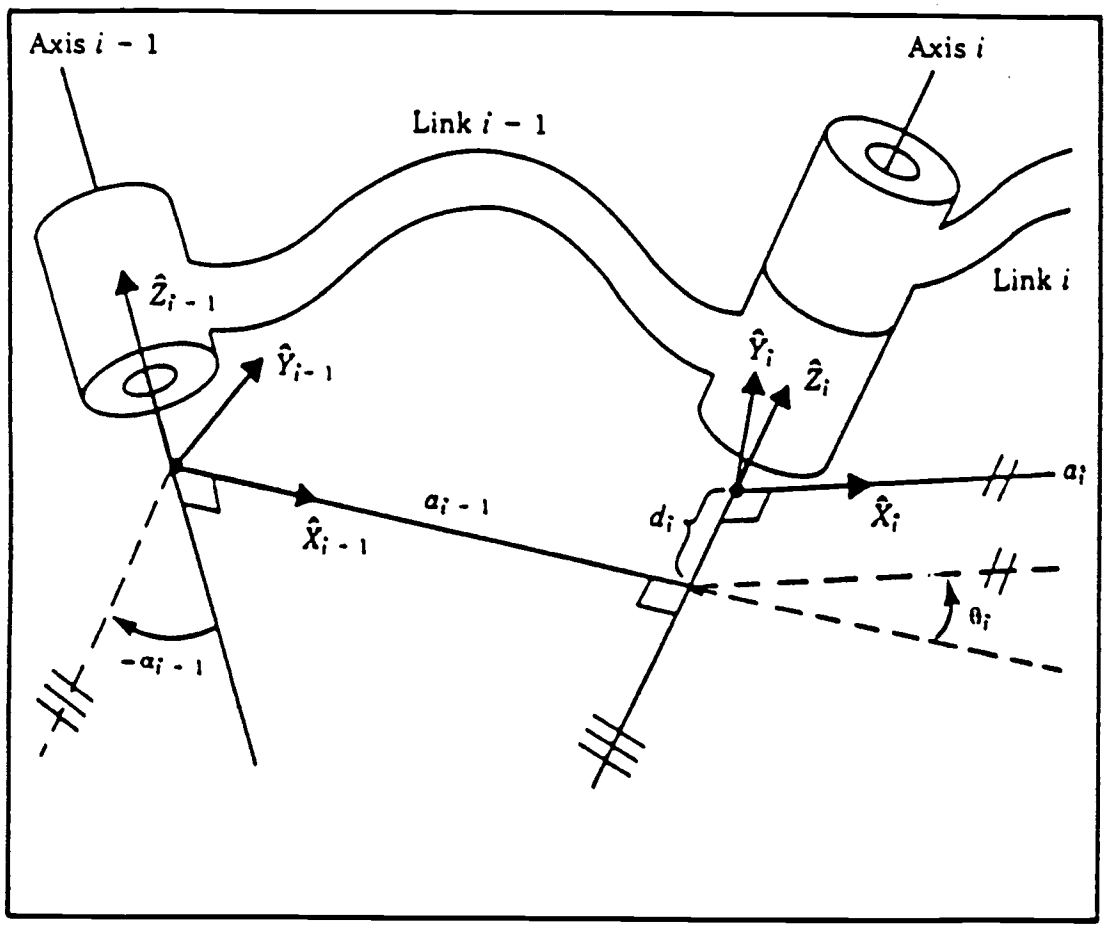


Figure 2.1: Parameters relating adjacent link coordinate systems.

$$
{}^{i-1}_{i}T = \begin{bmatrix} c\theta_i & -s\theta_i & 0 & a_{i-1} \\ s\theta_i c\alpha_{i-1} & c\theta_i c\alpha_{i-1} & -s\alpha_{i-1} & -s\alpha_{i-1}d_i \\ s\theta_i s\alpha_{i-1} & c\theta_i s\alpha_{i-1} & c\alpha_{i-1} & c\alpha_{i-1}d_i \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.2)
$$

The elements of the fourth column of each transform must give the coordinates of the origin of the next higher frame.

We can obtain the position and orientation of any frame with respect to any other frame. The position and orientation of frame {N} with respect to frame {0}:

$$
{}^{0}_{N}T = {}^{0}_{1}T\ {}^{1}_{2}T\ {}^{2}_{3}T \cdots {}^{N-2}_{N-1}T\ {}^{N-1}_{N}T \quad . \qquad (2.3)
$$

2.2.2 INVERSE KINEMATICS

Since trajectories can be planned in terms of the position and orientation of the end-effector, the position and orientation of the end-effector must be transformed into joint coordinate motions to derive the joint servo controllers. Inverse kinematics is used to find the corresponding joint angles of the manipulator so that the end-effector can be positioned as desired. All the points on the specified robot path and specified goal points must lie within the workspace for a solution to exist.

In general, the inverse kinematics problem can be solved by various methods, such as inverse transform, screw algebra, dual matrices, dual quaternian, numerical, and closed form (algebraic and geometric) approaches (Craig 1986; Fu et al. 1987). All robotics books give full details of some of these approaches. In our solution, we will

use the algebraic solution (Craig 1986) to transform the position and orientation of the end-effector on the specified robot path into joint coordinate motions.

We assume that the necessary transformations have been performed so that the each point on the specified robot path is a specification of the wrist frame relative to the base frame, ${}^{B}_{W}T$. Since we will work with 4-link SCARA robot that will be treated as a 2-link planar manipulator, specification of goal points may be easily accomplished by specifying x, y, and ψ, where ψ is the orientation of link 2 in the plane (relative to the $+\hat{X}_0$ axis). Thus, we will assume a transformation with the structure

$$ {}^{B}_{W}T = \begin{bmatrix} c_\psi & -s_\psi & 0 & x \\ s_\psi & c_\psi & 0 & y \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (2.4) $$

We must solve it for θ_1 and θ_2. Solution of this is explicitly shown in Section 2.4.

2.3 MANIPULATOR DYNAMICS

There are a number of ways of obtaining the dynamic equations of a robotic manipulator, i.e., the equations which relate joint forces and torques to positions, velocities, and accelerations. The Newton-Euler formulation of mechanism dynamics yields a set of differential equations which are easy to manipulate for robot control problems, and so will be used here.

The resulting Newton-Euler equations of motion are a set of forward (outward) and backward (inward) recursive equations. The forward recursion propagates kinematics information (linear velocities, angular velocities, linear accelerations, and angular accelerations at the center of mass of each link) from the inertial frame to end-effector frame. The backward recursion propagates the forces and moments exerted on each link from the manipulator end-effector frame to the manipulator base frame (Luh, Walker, and Paul 1980; Brady et al. 1982; Craig 1986; Fu et al. 1987).

We consider each link of a manipulator as a rigid body. Also, the location of the center of mass and the inertia tensor of the link are known; thus, its mass distribution is completely characterised.

2.3.1 ITERATIVE NEWTON-EULER DYNAMIC FORMULATION

We compute the torques that correspond to a given robot trajectory. We can compute the joint torques required to cause desired robot motion with knowledge of the kinematics and mass distribution information of the robot.

The iterative Newton-Euler dynamics algorithm for computing joint torques is composed of two parts. First, link velocities and accelerations are iteratively computed from link 1 out to link N and the Newton-Euler equations are applied to each link. Second, forces and torques of interaction and joint torques are computed recursively from link N back to link 1 (Luh, Walker, and Paul 1980; Brady et al. 1982; Craig 1986).

2.3.1.1 OUTWARD ITERATIONS

The outward (forward) iteration propagates angular velocities, angular accelerations, linear accelerations, inertial link forces, and inertial link torques from link 1 to link N moving successively, link by link. For rotational joint i+1, the recursive equations are:

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\ {}^{i}\omega_{i} + \dot{\theta}_{i+1}\ {}^{i+1}\hat{Z}_{i+1} \tag{2.5} $$

$$ {}^{i+1}\dot{\omega}_{i+1} = {}^{i+1}_{i}R\,{}^{i}\dot{\omega}_{i} + {}^{i+1}_{i}R\,{}^{i}\omega_{i} \times \dot{\theta}_{i+1}\ {}^{i+1}\hat{Z}_{i+1} + \theta_{i+1}\ {}^{i+1}\hat{Z}_{i+1} \tag{2.6} $$

$$ {}^{i+1}\dot{\vartheta}_{i+1} = {}^{i+1}_{i}R[\,{}^{i}\dot{\omega}_{i} \times {}^{i}P_{i+1} + {}^{i}\omega_{i} \times ({}^{i}\omega_{i} \times {}^{i}P_{i+1}) + {}^{i}\dot{\vartheta}_{i}] \tag{2.7} $$

$$ {}^{i+1}\dot{\vartheta}_{ci+1} = {}^{i+1}\dot{\omega}_{i+1} \times {}^{i+1}P_{ci+1} + {}^{i+1}\omega_{i+1} \times ({}^{i+1}\omega_{i+1} \times {}^{i+1}P_{ci+1}) + {}^{i+1}\dot{\vartheta}_{i+1} \tag{2.8} $$

$$ {}^{i+1}F_{i+1} = m_{i+1}\ {}^{i+1}\dot{\vartheta}_{ci+1} \tag{2.9} $$

$$ {}^{i+1}N_{i+1} = {}^{ci+1}I_{i+1}\ {}^{i+1}\dot{\omega}_{i+1} + {}^{i+1}\omega_{i+1} \times {}^{ci+1}I_{i+1}\ {}^{i+1}\omega_{i+1} \tag{2.10} $$

where

$i = 0,1,2,\ldots,5$,

${}^{i+1}\omega_{i+1}$ is the angular velocity of link i+1,

${}^{i+1}\dot{\omega}_{i+1}$ is the angular acceleration of link i+1,

${}^{i+1}\dot{\vartheta}_{i+1}$ is the linear acceleration of link i+1,

${}^{i+1}\dot{\vartheta}_{ci+1}$ is the linear acceleration of the center of mass of link i+1,

${}^{i+1}F_{i+1}$ is the inertial force acting at the center of mass of link i+1,

${}^{i+1}N_{i+1}$ is the inertial torque acting at the center of mass of link i+1,

${}^{ci+1}I_{i+1}$ is the inertia tensor of link i+1 about its center of mass,

${}^{i+1}_{i}R$ is the rotation matrix relating frames {i} and {i+1},

${}^{i}P_{i+1}$ is the vector from frame {i} to frame {i+1},

${}^{i+1}P_{ci+1}$ is the vector from frame {i+1} to link i+1 center of mass, and

m_{i+1} is the mass of link i+1.

The application of the equations to link 1 are simple since ${}^{0}\omega_0 = {}^{0}\dot{\omega}_0 = 0$. The effect of gravity loading on the links can be included simply by setting ${}^{0}\dot{v}_0 = G$, where G is the gravity vector.

2.3.1.2 INWARD ITERATIONS

The inward (backward) iteration propagates the forces and moments exerted on link i+1 by link i from the link N to the base of the robot. For rotational joint i, the recursive equations are:

$$ {}^{i}f_i = {}^{i}_{i+1}R \; {}^{i+1}f_{i+1} + {}^{i}F_i \,, \qquad (2.11) $$

$$ {}^{i}n_i = {}^{i}N_i + {}^{i}_{i+1}R \; {}^{i+1}n_{i+1} + {}^{i}P_{ci} \times {}^{i}F_i + {}^{i}P_{i+1} \times {}^{i}_{i+1}R \; {}^{i+1}f_{i+1} \,, \qquad (2.12) $$

$$ \tau_i = {}^{i}n_i^{T} \; {}^{i}\hat{Z}_i \,, \qquad (2.13) $$

where

$i = 6,5,\ldots,1$,

${}^{i}f_{i}$ is the force exerted on link i by link i-1,

${}^{i}n_{i}$ is the torque exerted on link i by link i-1,

τ_i is the input torque at joint i.

For a manipulator moving in free space , ${}^{N+1}f_{N+1}$ and ${}^{N+1}n_{N+1}$ are set equal to zero. In this formulation the implicit reference frame is the base frame.

2.3.2 MOTION EQUATIONS OF THE MANIPULATOR

When the Newton-Euler equations are evaluated symbolically for any manipulator, they yield an equation of motion for the manipulator which can be written as:

$$\tau = M(\underline{\theta})\ddot{\underline{\theta}} + V(\underline{\theta},\dot{\underline{\theta}}) + G(\underline{\theta}) + F(\underline{\theta},\dot{\underline{\theta}}) \qquad (2.14)$$

where

$M(\underline{\theta})$ is the NxN mass matrix of the manipulator,

$V(\underline{\theta},\dot{\underline{\theta}})$ is an Nx1 vector defining Coriolis and centrifugal terms,

$G(\underline{\theta})$ is an Nx1 vector defining the gravity terms,

$F(\underline{\theta},\dot{\underline{\theta}})$ is an NxN friction matrix,

τ is an Nx1 vector of input generalized torques,

$\underline{\theta}$ is an Nx1 vector of joint displacements of the manipulator,

$\dot{\underline{\theta}}$ is an Nx1 vector of joint velocity of the manipulator, and

$\ddot{\underline{\theta}}$ is an Nx1 vector of joint acceleration of the manipulator.

Since the term $V(\underline{\theta},\dot{\underline{\theta}})$ in Equ. (2.14) has both position and velocity dependence, we call Equ. (2.14) a **State Space Equation.** Each element of

$M(\underline{\theta})$ and $G(\underline{\theta})$ is a complex function of $\underline{\theta}$. In other words, manipulator mass matrix, $M(\underline{\theta})$ as well as the gravitational force is dependent on the configuration of the manipulator. Each element of $V(\underline{\theta},\dot{\underline{\theta}})$ and $F(\underline{\theta},\dot{\underline{\theta}})$ is a complex function of both $\underline{\theta}$ and $\dot{\underline{\theta}}$. The Coriolis forces depend on the product of two different joint velocities; the centrifugal forces depend on the square of a joint velocity.

By writing the velocity dependent term, $V(\underline{\theta},\dot{\underline{\theta}})$, in a different form, the equation of motion for the manipulator can be written as (Craig 1986):

$$\tau = M(\underline{\theta})\ddot{\underline{\theta}} + B(\underline{\theta})[\dot{\underline{\theta}}\ \dot{\underline{\theta}}] + C(\underline{\theta})[\dot{\underline{\theta}}^2] + G(\underline{\theta}) \qquad (2.15)$$

where

$B(\underline{\theta})$ is a matrix of dimensions N x N(N-1)/2 of Coriolis coefficients,

$C(\underline{\theta})$ is an N x N matrix of centrifugal coefficients,

$[\dot{\underline{\theta}}\ \dot{\underline{\theta}}]$ is an N(N-1)/2 x 1 vector of joint velocity products given by

$$[\dot{\underline{\theta}}\ \dot{\underline{\theta}}] = [\ \dot{\theta}_1\dot{\theta}_2\ \dot{\theta}_1\dot{\theta}_3\ \ldots\ \dot{\theta}_{N-1}\dot{\theta}_N\ ]^T, \qquad (2.16)$$

$[\dot{\underline{\theta}}^2]$ is an Nx1 vector of square of joint velocity given by

$$[\dot{\underline{\theta}}^2] = [\ \dot{\theta}_1^2\ \dot{\theta}_2^2\ \ldots\ \dot{\theta}_N^2\ ]^T . \qquad (2.17)$$

Since the matrices are only functions of manipulator position, the Equ. (2.15) is called the **Configuration Space Equation.**

2.4 EXAMPLE OF THE 4-LINK SCARA ROBOT

In this section we work out the 4-link SCARA robot with four degrees of freedom shown in Figure 2.2. We will compute the inverse kinematics and the closed form dynamic equations for this SCARA robot. Two motors are located at the joints, which produce the horizontal link motion. The other two motors are on the forearm: one produces a translation motion along the vertical axis, while the other rotates the gripper about the vertical axis. Since the position and orientation of the gripper throughout the motion are about the vertical axis, we will consider only the horizontal link motion in xy-plane. For simplicity, we assume that the mass distribution is uniform. In other words, we have a center of mass at the mid-point of each link. These masses are m_1 and m_2. Also, there is a mass, m_3, which exists at the distal end of link 2 for mass of link 3, mass of gripper, and payload mass. Since link 3 has zero length and link mass m_3, we may consider mass m_3 as a point mass at the distal end of link 2.

Since we consider only the horizontal link motion of SCARA robot in xy-plane, we may treat it as a two degree-of-freedom planar manipulator. Figure 2.3 shows the two degree-of-freedom planar manipulator.

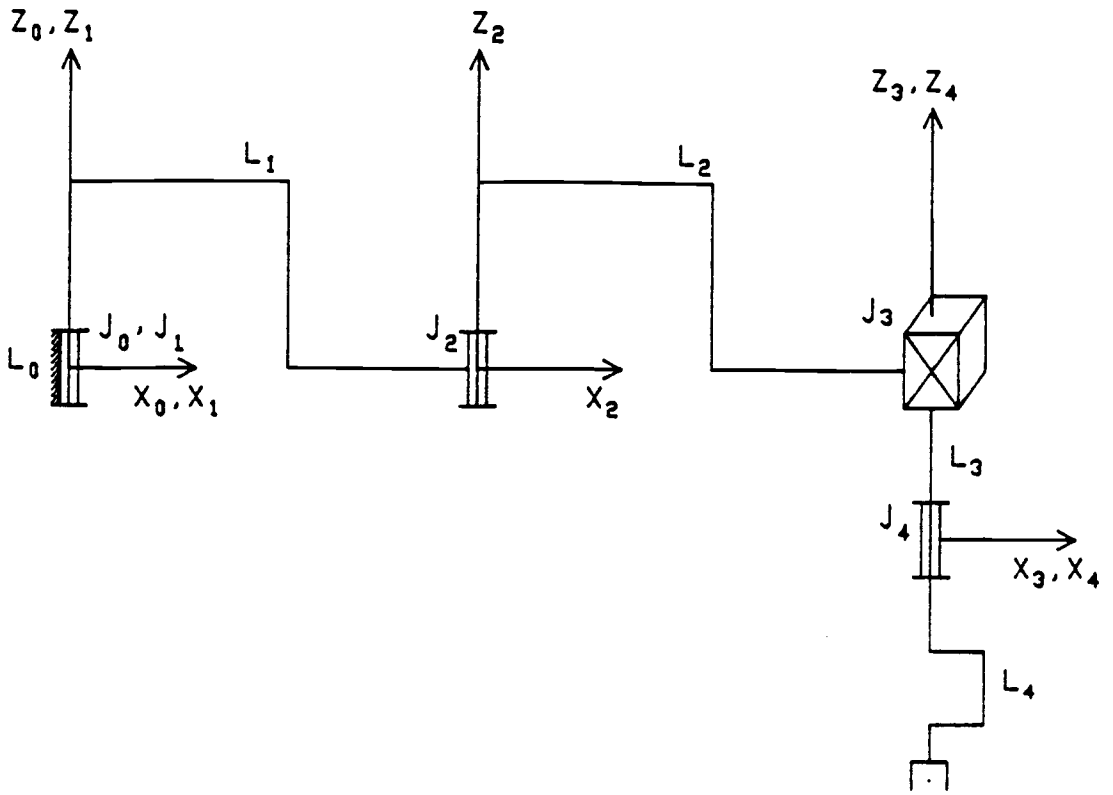


Figure 2.2: Coordinate frames for the 4 DOF SCARA robot.

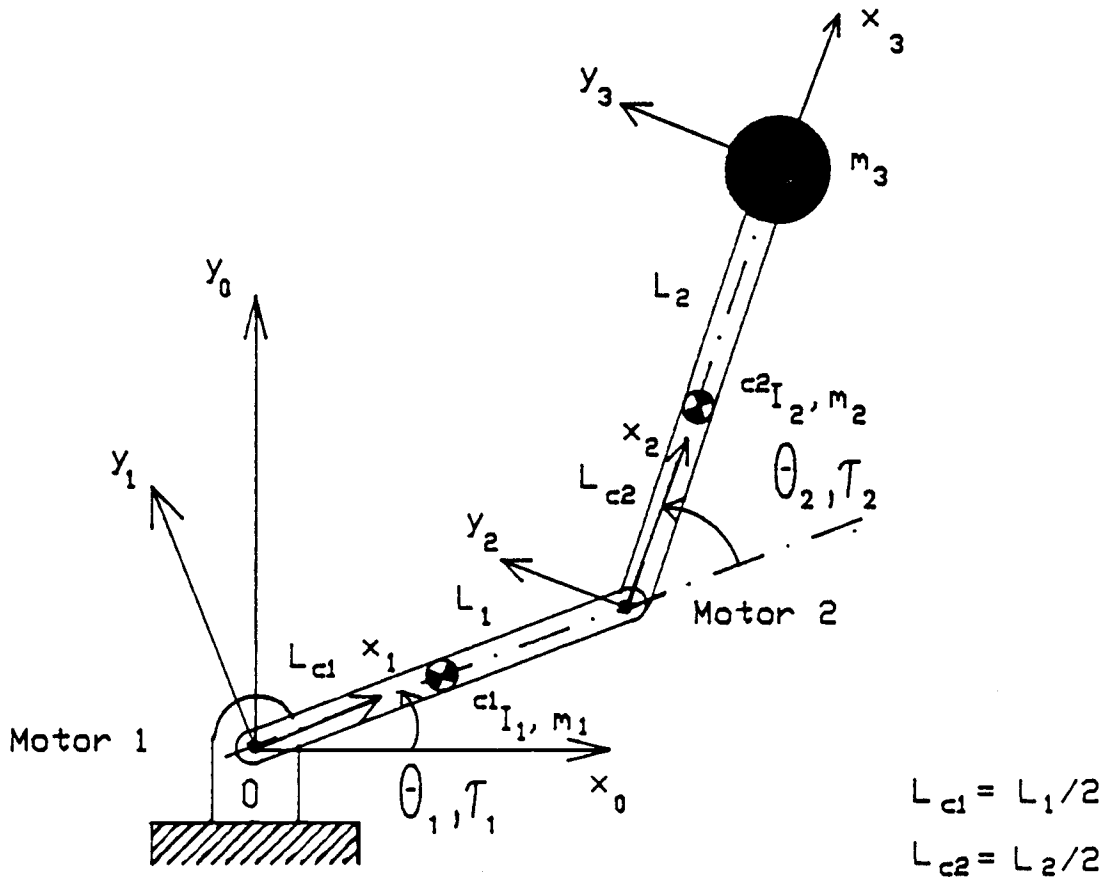


Figure 2.3: Dimensions and mass properties of 2 DOF planar manipulator.

2.4.1 INVERSE MANIPULATOR KINEMATICS

As an example of the algebraic solution technique applied to the two degree of freedom planar manipulator, we will solve the kinematic equations of the SCARA robot. The link parameters of the 4-link SCARA robot are shown in Figure 2.4.

Link	a_{i-1}	α_{i-1}	d_i	θ_i
1	0	0	0	$\theta_1(t)$
2	l_1	0	0	$\theta_2(t)$
3	l_2	0	$d_3(t)$	0
4	0	0	0	$\theta_4(t)$

Figure 2.4: Link parameters of the SCARA robot.

We compute each of the link transformations:

$$ {}^0_1T = \begin{bmatrix} c_1 & -s_1 & 0 & 0 \\ s_1 & c_1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , $$

$$ {}^1_2T = \begin{bmatrix} c_2 & -s_2 & 0 & l_1 \\ s_2 & c_2 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \qquad (2.18) $$

$$ {}^{2}_{3}T = \begin{bmatrix} 1 & 0 & 0 & l_2 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} , $$

$$ {}^{3}_{4}T = \begin{bmatrix} c_4 & -s_4 & 0 & 0 \\ s_4 & c_4 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} . $$

where c_1 is shorthand for $\cos\theta_1$, s_1 for $\sin\theta_1$, and so on.

Since we treat the 4-link SCARA robot as a two degree-of-freedom planar robot arm with rotational joints as shown in Figure 2.3, we wish to form the transformation matrix ${}^{0}_{3}T$ to calculate the position and orientation of the tip of the robot arm. Frame {3} has been attached at the end of the two-link manipulator. We now form ${}^{0}_{3}T$ by matrix multiplication of the individual link matrices as follows:

$$ {}^{1}_{3}T = {}^{1}_{2}T \, {}^{2}_{3}T = \begin{bmatrix} c_2 & -s_2 & 0 & l_1+l_2c_2 \\ s_2 & c_2 & 0 & l_2s_2 \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} , \qquad (2.19) $$

$$ {}^{0}_{3}T = {}^{0}_{1}T\,{}^{1}_{3}T = \begin{bmatrix} c_{12} & -s_{12} & 0 & l_1c_1+l_2c_{12} \\ s_{12} & c_{12} & 0 & l_1s_1+l_2s_{12} \\ 0 & 0 & 1 & d_3 \\ 0 & 0 & 0 & 1 \end{bmatrix} . \qquad (2.20) $$

To focus our discussion on inverse kinematics, we will assume that the necessary transformations have been performed so that the Cartesian end-effector path points are specification of the wrist frame relative to the base frame, that is, ${}^{B}_{W}T$. We will assume a transformation with the structure

$$ {}^{B}_{W}T = \begin{bmatrix} c_\psi & -s_\psi & 0.0 & x \\ s_\psi & c_\psi & 0.0 & y \\ 0.0 & 0.0 & 1.0 & 0.0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \qquad (2.21) $$

where ψ is the orientation of link 2 in the plane. By equating (2.20) and (2.21) we obtain a set of nonlinear equations which must be solved for θ_1 and θ_2:

$$ c_\psi = c_{12} \qquad (2.22) $$

$$ s_\psi = s_{12} \qquad (2.23) $$

$$ x = l_1c_1 + l_2c_{12} \qquad (2.24) $$

$$ y = l_1s_1 + l_2s_{12} \qquad (2.25) $$

$$ d_3 = 0.0 \qquad (2.26) $$

If we square both (2.24) and (2.25) and add them, we obtain

$$ x^2 + y^2 = l_1^2 + l_2^2 + 2l_1l_2c_2, \qquad (2.27) $$

where

$$c_{12} = c_1c_2 - s_1s_2$$
$$s_{12} = c_1s_2 + s_1c_2 \quad . \tag{2.28}$$

Solving (2.27) for c_2 we obtain

$$c_2 = \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2} \quad . \tag{2.29}$$

In order for a solution to exist, the right hand side of (2.29) must have a value between -1 and 1. If this constraint is not satisfied, then the goal point is too far away for the manipulator to reach. We can write an expression for s_2 as

$$s_2 = \pm \sqrt{(1-c_2^2)} \tag{2.30}$$

The choice of signs in (2.30) corresponds to the multiple solution in which we can choose the "elbow-up" or the "elbow-down" solution. We compute θ_2 as

$$\theta_2 = \text{Atan2}(s_2,c_2). \tag{2.31}$$

After finding θ_2, we can solve (2.24) and (2.25) for θ_1. We can write (2.24) and (2.25) in the form

$$k_1c_1 - k_2s_1 = x \tag{2.32}$$
$$k_1s_1 + k_2c_1 = y \tag{2.33}$$

where

$$k_1 = l_1 + l_2c_2$$
$$k_2 = l_2s_2 \tag{2.33a}$$

Since we have two equations (2.32 and 2.33) and two unknowns (c_1 and s_1), solving (2.32) and (2.33) for c_1 and s_1 we obtain

$$c_1 = \frac{k_1 x + k_2 y}{k_1^2 + k_2^2}, \tag{2.34}$$

$$s_1 = \frac{k_1 y - k_2 x}{k_1^2 + k_2^2}. \tag{2.35}$$

Using the two-argument arctangent, we get

$$\theta_1 = \text{Atan2}(s_1, c_1). \tag{2.36}$$

2.4.2 CLOSED FORM DYNAMIC EQUATIONS

Here we compute the closed form dynamic equations for the 2-link planar manipulator shown in Figure 2.3. Let us obtain the Newton-Euler equations of motion for the two individual links, and then derive the closed-form dynamic equations in terms of joint displacements θ_1 and θ_2, and joint torques τ_1 and τ_2. We assume that the centroid of link i is located on the center line passing through adjacent joints at a distance l_{ci} from joint i, as shown in Figure 2.3.

First, we must determine the value of the various quantities which will appear in the recursive Newton-Euler equations. The vectors which locate the center of mass for each link are

$$^1P_{c1} = \begin{bmatrix} l_{c1} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} l_1/2 \\ 0 \\ 0 \end{bmatrix},$$

$$^2P_{c2} = \begin{bmatrix} l_{c2} \\ 0 \\ 0 \end{bmatrix} = \begin{bmatrix} l_2/2 \\ 0 \\ 0 \end{bmatrix},$$

$$ {}^{3}P_{c3} = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} . $$

Since we assume uniform mass distribution, we have to calculate the inertia tensor written at the center of mass for each link. We assume that there are no forces acting on the end-effector, so we have

$$ f_4 = 0 \quad \text{and} \quad n_4 = 0. $$

Since the base of the robot is not rotating, we have

$$ {}^{0}\omega_0 = 0, \; {}^{0}\dot{\omega}_0 = 0, \text{ and } {}^{0}\vartheta_0 = 0 . $$

To include gravity forces we will use

$$ {}^{0}\dot{\vartheta}_0 = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} . $$

The rotation between successive link frames is given by

$$ {}^{i}_{i+1}R = \begin{bmatrix} c_{i+1} & -s_{i+1} & 0 \\ s_{i+1} & c_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} , $$

$$ {}^{i+1}_{1}R = \begin{bmatrix} c_{i+1} & s_{i+1} & 0 \\ -s_{i+1} & c_{i+1} & 0 \\ 0 & 0 & 1 \end{bmatrix} . $$

Let us apply Equations (2.5) through (2.13).

The outward iterations for link 1 are as follows:

$$ {}^{1}\omega_1 = \dot{\theta}_1 \, {}^{1}\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} , $$

$$ {}^{1}\dot{\omega}_1 = \ddot{\theta}_1 \, {}^{1}\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1 \end{bmatrix} , $$

$$ {}^{1}\dot{v}_1 = \begin{bmatrix} c_1 & s_1 & 0 \\ -s_1 & c_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} , $$

$$ {}^{1}\dot{v}_{c1} = \begin{bmatrix} 0 \\ l_{c1}\ddot{\theta}_1 \\ 0 \end{bmatrix} + \begin{bmatrix} -l_{c1}\dot{\theta}_1^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ 0 \\ g \end{bmatrix} = \begin{bmatrix} -l_{c1}\dot{\theta}_1^2 \\ l_{c1}\ddot{\theta}_1 \\ g \end{bmatrix} , $$

$$ {}^{1}F_1 = \begin{bmatrix} -m_1 l_{c1}\dot{\theta}_1^2 \\ m_1 l_{c1}\ddot{\theta}_1 \\ m_1 g \end{bmatrix} = \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \end{bmatrix} , $$

$$ {}^{1}N_1 = \begin{bmatrix} 0 \\ 0 \\ I_{z1}\ddot{\theta}_1 \end{bmatrix} . \qquad (2.37a\text{-}f) $$

The outward iterations for link 2 are as follows:

$$^2\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix} ,$$

$$^2\dot{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 \end{bmatrix} ,$$

$$^2\dot{v}_2 = \begin{bmatrix} c_2 & s_2 & 0 \\ -s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} -l_1\dot{\theta}_1^2 \\ l_1\ddot{\theta}_1 \\ g \end{bmatrix} = \begin{bmatrix} l_1 s_2\ddot{\theta}_1 - l_1 c_2\dot{\theta}_1^2 \\ l_1 c_2\ddot{\theta}_1 + l_1 s_2\dot{\theta}_1^2 \\ g \end{bmatrix}$$

$$^2\dot{v}_{c2} = \begin{bmatrix} 0 \\ l_{c2}(\ddot{\theta}_1+\ddot{\theta}_2) \\ 0 \end{bmatrix} + \begin{bmatrix} -l_{c2}(\dot{\theta}_1+\dot{\theta}_2)^2 \\ 0 \\ 0 \end{bmatrix} + {}^2\dot{v}_2$$

$$= \begin{bmatrix} l_1 s_2\ddot{\theta}_1 - l_1 c_2\dot{\theta}_1^2 - l_{c2}(\dot{\theta}_1+\dot{\theta}_2)^2 \\ l_1 c_2\ddot{\theta}_1 + l_1 s_2\dot{\theta}_1^2 + l_{c2}(\ddot{\theta}_1+\ddot{\theta}_2) \\ g \end{bmatrix} ,$$

$$^2F_2 = \begin{bmatrix} m_2 l_1 s_2\ddot{\theta}_1 - m_2 l_1 c_2\dot{\theta}_1^2 - m_2 l_{c2}(\dot{\theta}_1+\dot{\theta}_2)^2 \\ m_2 l_1 c_2\ddot{\theta}_1 + m_2 l_1 s_2\dot{\theta}_1^2 + m_2 l_{c2}(\ddot{\theta}_1+\ddot{\theta}_2) \\ m_2 g \end{bmatrix} = \begin{bmatrix} F_{21} \\ F_{22} \\ F_{23} \end{bmatrix} ,$$

$$ {}^{2}N_2 = \begin{bmatrix} 0 \\ 0 \\ I_{z2}(\ddot{\theta}_1+\ddot{\theta}_2) \end{bmatrix} . \qquad (2.38a\text{-}f) $$

The outward iterations for link 3 are as follows:

Since the frame {3}, which is parallel to the frame {2}, is attached at the end of the manipulator as shown in Figure 2.3, we will have the same orientation (rotation) for the link 3. Hence we have

$$ {}^{3}\omega_3 = {}^{2}\omega_2 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix} , $$

$$ {}^{3}\dot{\omega}_3 = {}^{2}\dot{\omega}_2 = \begin{bmatrix} 0 \\ 0 \\ \ddot{\theta}_1+\ddot{\theta}_2 \end{bmatrix} , $$

$$ {}^{3}\dot{v}_3 = \begin{bmatrix} 0 \\ l_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ 0 \end{bmatrix} + \begin{bmatrix} -l_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ 0 \\ 0 \end{bmatrix} + \begin{bmatrix} l_1 s_2\ddot{\theta}_1 - l_1 c_2\dot{\theta}_1^2 \\ l_1 c_2\ddot{\theta}_1 + l_1 s_2\dot{\theta}_1^2 \\ g \end{bmatrix} $$

$$ = \begin{bmatrix} l_1 s_2\ddot{\theta}_1 - l_1 c_2\dot{\theta}_1^2 - l_2(\dot{\theta}_1+\dot{\theta}_2)^2 \\ l_1 c_2\ddot{\theta}_1 + l_1 s_2\dot{\theta}_1^2 + l_2(\ddot{\theta}_1+\ddot{\theta}_2) \\ g \end{bmatrix} , $$

$$^3\dot{v}_{c3} = {}^3\dot{v}_3 \ ,$$

$$^3F_3 = \begin{bmatrix} m_3 l_1 s_2 \ddot{\theta}_1 - m_3 l_1 c_2 \dot{\theta}_1^2 - m_3 l_2 (\dot{\theta}_1 + \dot{\theta}_2)^2 \\ m_3 l_1 c_2 \ddot{\theta}_1 + m_3 l_1 s_2 \dot{\theta}_1^2 + m_3 l_2 (\ddot{\theta}_1 + \ddot{\theta}_2) \\ m_3 g \end{bmatrix} = \begin{bmatrix} F_{31} \\ F_{32} \\ F_{33} \end{bmatrix} \ ,$$

$$^3N_3 = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \ . \qquad (2.39a\text{-}f)$$

The inward iterations for link 3 are as follows:

$$^3f_3 = {}^3F_3$$

$$^3n_3 = {}^3N_3 \ . \qquad (2.40a\text{-}b)$$

The inward iterations for link 2 are as follows:

$$^2f_2 = {}^2F_2 + {}^3F_3 = \begin{bmatrix} f_{21} \\ f_{22} \\ f_{23} \end{bmatrix}$$

$$= \begin{bmatrix} (m_2+m_3) l_1 s_2 \ddot{\theta}_1 - (m_2+m_3) l_1 c_2 \dot{\theta}_1^2 - (m_2 l_{c2} + m_3 l_2)(\dot{\theta}_1 + \dot{\theta}_2)^2 \\ (m_2+m_3) l_1 c_2 \ddot{\theta}_1 + (m_2+m_3) l_1 s_2 \dot{\theta}_1^2 + (m_2 l_{c2} + m_3 l_2)(\ddot{\theta}_1 + \ddot{\theta}_2)^2 \\ (m_2+m_3) g \end{bmatrix}$$

$$
{}^{2}n_{2} = \begin{bmatrix} 0 \\ 0 \\ I_{z2}(\ddot{\theta}_1+\ddot{\theta}_2) \end{bmatrix} + \begin{bmatrix} 0 \\ -l_{c2}F_{23} \\ l_{c2}F_{22} \end{bmatrix} + \begin{bmatrix} 0 \\ -l_2F_{33} \\ l_2F_{32} \end{bmatrix}
$$

$$
= \begin{bmatrix} 0 \\ -l_{c2}F_{23}-l_2F_{33} \\ I_{z2}(\ddot{\theta}_1+\ddot{\theta}_2)+l_{c2}F_{22}+l_2F_{32} \end{bmatrix} = \begin{bmatrix} n_{21} \\ n_{22} \\ n_{23} \end{bmatrix} . \qquad (2.41a\text{-}b)
$$

The inward iterations for link 1 are as follows:

$$
{}^{1}f_{1} = \begin{bmatrix} c_2 & -s_2 & 0 \\ s_2 & c_2 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} f_{21} \\ f_{22} \\ f_{23} \end{bmatrix} + \begin{bmatrix} F_{11} \\ F_{12} \\ F_{13} \end{bmatrix}
$$

$$
= \begin{bmatrix} F_{11}+f_{21}c_2-f_{22}s_2 \\ F_{12}+f_{21}s_2+f_{22}c_2 \\ F_{13}+f_{23} \end{bmatrix} ,
$$

$$
{}^{1}n_{1} = \begin{bmatrix} 0 \\ 0 \\ I_{z1}\ddot{\theta}_1 \end{bmatrix} + \begin{bmatrix} n_{21}c_2-n_{22}s_2 \\ n_{21}s_2+n_{22}c_2 \\ n_{23} \end{bmatrix} + \begin{bmatrix} 0 \\ -l_{c1}F_{13} \\ l_{c1}F_{12} \end{bmatrix} + \begin{bmatrix} 0 \\ -l_1f_{23} \\ l_1s_2f_{21}+l_1c_2f_{22} \end{bmatrix}
$$

$$
= \begin{bmatrix} n_{21}c_2-n_{22}s_2 \\ n_{21}s_2+n_{22}c_2-l_{c1}F_{13}-l_1f_{23} \\ n_{23}+l_{c1}F_{12}+l_1s_2f_{21}+l_1c_2f_{22}+I_{z1}\ddot{\theta}_1 \end{bmatrix} . \qquad (2.42a\text{-}b)
$$

Extracting the $\hat{Z}$ components of the ${}^{i}n_{i}$, we find the joint torques:

$$\tau_1 = (m_1 l_{c1}^2 + m_2 l_1^2 + m_2 l_{c2}^2 + 2m_2 l_1 l_{c2} c_2 + m_3 l_1^2 + m_3 l_2^2 + 2m_3 l_1 l_2 c_2 + I_{z1} + I_{z2})\ddot{\theta}_1$$
$$+ (m_2 l_{c2}^2 + m_2 l_1 l_{c2} c_2 + m_3 l_2^2 + m_3 l_1 l_2 c_2 + I_{z2})\ddot{\theta}_2$$
$$- (m_2 l_1 l_{c2} s_2 + m_3 l_1 l_2 s_2)\dot{\theta}_2^2 - (2m_2 l_1 l_{c2} s_2 + 2m_3 l_1 l_2 s_2)\dot{\theta}_1\dot{\theta}_2 \quad ,$$

$$\tau_2 = (m_2 l_1 l_{c2} c_2 + m_2 l_{c2}^2 + m_3 l_2^2 + m_3 l_1 l_2 c_2 + I_{z2})\ddot{\theta}_1$$
$$+ (m_2 l_{c2}^2 + m_3 l_2^2 + I_{z2})\ddot{\theta}_2 + (m_2 l_1 l_{c2} s_2 + m_3 l_1 l_2 s_2)\dot{\theta}_1^2 \quad ,$$

$$\tau_3 = 0 \quad . \qquad (2.43a\text{-}c)$$

Equations (2.43) give expressions for the joint torques at the actuators as a function of joint positions, velocities, and accelerations. Equations (2.43) can be written in the state space equation form as follows:

$$\tau_1 = \tau_{11}\ddot{\theta}_1 + \tau_{12}\ddot{\theta}_2 + \tau_{13}\dot{\theta}_2^2 + \tau_{14}\dot{\theta}_1\dot{\theta}_2 \quad ,$$

$$\tau_2 = \tau_{21}\ddot{\theta}_1 + \tau_{22}\ddot{\theta}_2 + \tau_{23}\dot{\theta}_1^2 \quad . \qquad (2.44a\text{-}b)$$

$$\underline{\tau} = \begin{bmatrix} \tau_1 \\ \tau_2 \end{bmatrix} = \begin{bmatrix} \tau_{11} & \tau_{12} \\ \tau_{21} & \tau_{22} \end{bmatrix} \begin{bmatrix} \ddot{\theta}_1 \\ \ddot{\theta}_2 \end{bmatrix} + \begin{bmatrix} 0 \\ \tau_{23} \end{bmatrix} \dot{\theta}_1^2 + \begin{bmatrix} \tau_{13} \\ 0 \end{bmatrix} \dot{\theta}_2^2 + \begin{bmatrix} \tau_{14} \\ 0 \end{bmatrix} \dot{\theta}_1\dot{\theta}_2 \qquad (2.45)$$

Since we know ${}^{3}\dot{v}_3$ from manipulator dynamics, we can express the accelerations of the tip of the manipulator with respect to the nonmoving base frame by rotating them with the rotation matrix ${}^{0}_{3}R$. This rotation yields

$$
{}^0\dot{\vartheta}_3 = {}^0_3R\,{}^3\dot{\vartheta}_3
$$

$$
= \begin{bmatrix} -l_1 s_1 \ddot{\theta}_1 - l_2 s_{12}(\ddot{\theta}_1+\ddot{\theta}_2) - l_1 c_1 \dot{\theta}_1^2 - l_2 c_{12}(\dot{\theta}_1+\dot{\theta}_2)^2 \\ l_1 c_1 \ddot{\theta}_1 + l_2 c_{12}(\ddot{\theta}_1+\ddot{\theta}_2) - l_1 s_1 \dot{\theta}_1^2 - l_2 s_{12}(\dot{\theta}_1+\dot{\theta}_2)^2 \\ g \end{bmatrix} . \quad (2.46)
$$

2.5 REMARKS

Manipulator kinematics and dynamics are presented above for the derivation of motion equations for the manipulators. The method is very efficient, especially for robot dynamic simulation. A convention for link parameters and link frames is given in Appendix A. The manipulator Jacobian is also given in Appendix B.

CHAPTER 3

TRAJECTORY GENERATION

3.1 INTRODUCTION

In this chapter, we are concerned with methods of computing a trajectory in 2-dimensional space which describes the desired motion of a manipulator. We will describe motions of a manipulator as motions of the tip of the robot arm (of the end-effector frame), relative to the nonmoving robot base frame.

Robot trajectory planning is an important off-line stage. An executable robot trajectory must require the robot arm to move to a point inside robot workspace or move with a velocity, acceleration and joint torque that is physically possible. A sequence of desired intermediate points between the initial and final positions must be given in order to specify the robot motion in more detail. Each of these end-effector path points is a frame in terms of a desired position and orientation of the end-effector frame relative to the nonmoving robot base frame in Cartesian space to easily visualize the correct end-effector configurations. Since we initially generate the robot path in the Cartesian space, it is very easy to check the end-effector path which must avoid the obstacles in the workspace. Thus, we will call this path a geometric collision-free robot path. However, it may not prevent the manipulator's joints from colliding with the obstacles in the workspace of the manipulator.

Because of some disadvantages of the Cartesian space path planning, the joint space path planning method, which basically converts the Cartesian points on the robot path into their corresponding joint

coordinates and uses polynomial functions to interpolate these joint points, is used in this study. The joint space path planning method has the advantages of being computationally faster and makes it easier to deal with the manipulator dynamics constraints (Fu et al. 1987). In practice, the spatial paths are given in Cartesian coordinates. While it is generally difficult to convert a curve in Cartesian coordinates to that in joint coordinates, it is relatively easy to perform the conversion for individual points on the path. Therefore, we simply generate a number of points on the Cartesian path, convert them into joint coordinates, and use an interpolating curve technique to obtain a similar path in joint space.

To design Cartesian robot paths, we use interpolating and approximation curve techniques which are currently used in Computer-Aided Design as interactive design tools. These curve techniques are parabolic blending, Bezier, and B-spline curves. Bezier and B-spline curves have the advantage that they allow easy manipulation of the path's shape while requiring only a limited number of parameters. After transforming the Cartesian path points into joint space by applying inverse manipulator kinematics, cubic spline functions are used to interpolate these corresponding joint knot points for constructing the joint trajectories for robotic manipulators.

This chapter is mainly divided into four sections: Introduction, Cartesian Path Generation, Joint Trajectory Generation, and Remarks.

3.2 CARTESIAN PATH GENERATION

In this study, parabolic blending, Bezier, and B-spline curve techniques are used to construct Cartesian robot paths for the tip of the robotic manipulator or the end-effector of robot arm. It should be noted that Bezier and B-spline curves have the advantage that they allow easy manipulation of the path's shape.

3.2.1 PARABOLIC BLENDING CURVES

The technique for parabolic blending presented here is an "interpolation scheme which considers four consecutive points simultaneously. A smooth curve between the two interior points is generated by blending two overlapping parabolic segments. The first parabolic segment is defined by the first three points, and the last three points of the set of four points define the second parabolic segment. Each parabola which goes through three points is defined relative to its own local coordinate system. In general, a parabola can be completely specified by two end points and a third point on the curve" (Rogers and Adams 1976).

Two overlapping parabolas **P**(r) and **Q**(s) between four consecutive points in space are shown in Figure 3.1 (Rogers and Adams 1976). The position vectors P_0, P_1, P_2, and P_3 are defined in the Cartesian xyz-coordinate system while the blending parabolas **P**(r) and **Q**(s) are defined in a local coordinate system.

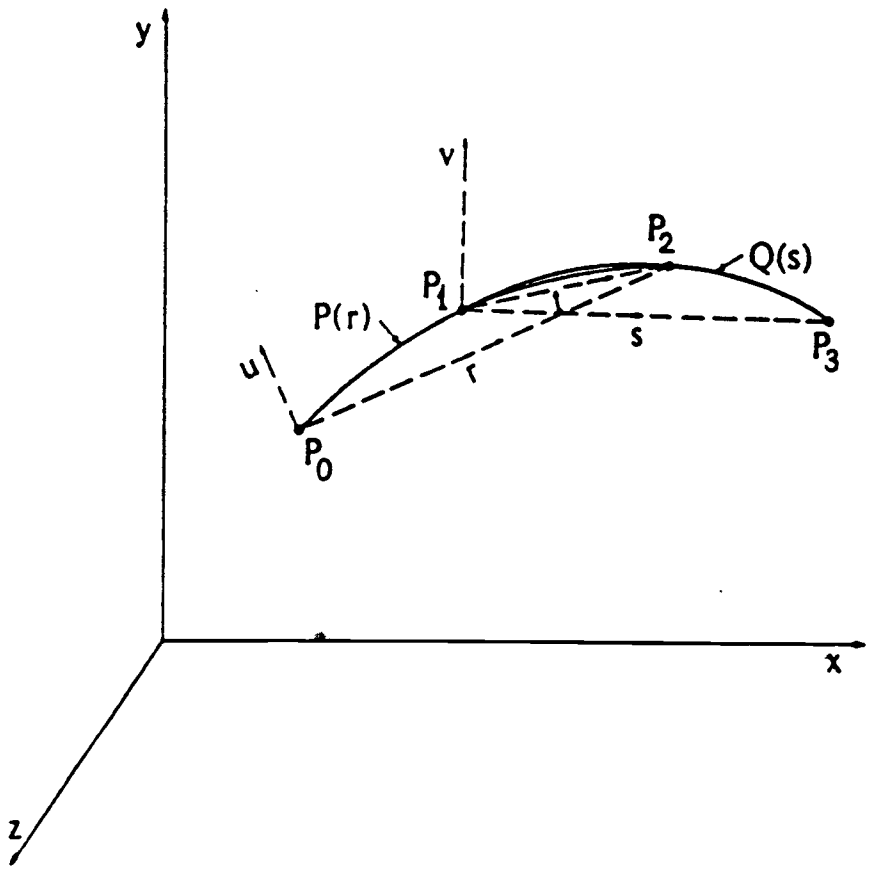


Figure 3.1: Parabolic blending.

A curve $\mathbf{C}(t)$ is a blend of the two overlapping parabolas. It is constructed between P_1 and P_2. The blending curve $\mathbf{C}(t)$ is given by

$$\mathbf{C}(t) = [1 - (\frac{t}{t_0})]\mathbf{P}(r) + [\frac{t}{t_0}]\mathbf{Q}(s) \qquad (3.1)$$

where t_0 is the distance between P_1 and P_2. The coefficients of $\mathbf{P}(r)$ and $\mathbf{Q}(s)$ act as blending functions and vary linearly between 1 and 0, and 0 and 1, respectively. The parabola $\mathbf{P}(r)$, which goes through P_0, P_1 and P_2, is given by the following equation

$$\mathbf{P}(r) = P_0 + \frac{r}{d}(P_2 - P_1) + \alpha r(d - r)[(P_1 - P_0) - x(P_2 - P_0)] \qquad (3.2)$$

where d is the chord length between P_0 and P_2. The value of the constant α is chosen such that the parabola $\mathbf{P}(r)$ passes through P_1,

$$\alpha = 1/[d^2x(1-x)] \; . \qquad (3.3)$$

Parametric equation for $r = r(t)$ is given as

$$r = xd + t\cos\theta \qquad (3.4)$$

where

$$\cos\theta = (\frac{P_2 - P_1}{t_0}) \cdot (\frac{P_2 - P_0}{d}) \; . \qquad (3.5)$$

Equation for x is given as

$$x = \frac{(P_1 - P_0) \cdot (P_2 - P_0)}{d^2} \; . \qquad (3.6)$$

In a similar manner, the parabola $\mathbf{Q}(s)$ is defined and passes through the points P_1, P_2, and P_3. The equation can be written as

$$\mathbf{Q}(s) = P_1 \frac{s}{e}(P_3 - P_1) + \beta s(e - s)[(P_2 - P_1) - x(P_3 - P_1)] \qquad (3.7)$$

where e is the chord length between P_1 and P_3. The value of β is chosen such that the parabola Q(s) passes through P_2,

$$\beta = 1/[e^2x(1-x)] \quad . \tag{3.8}$$

Parametric equation for s = s(t) is given as

$$s = t\cos\theta = t[(\frac{P_2 - P_1}{t_0})\cdot(\frac{P_3 - P_1}{e})] \quad . \tag{3.9}$$

Equation for x is given as

$$x = \frac{(P_2 - P_1)\cdot(P_3 - P_1)}{e^2} \quad . \tag{3.10}$$

A blending curve $C_i(t_i)$ is generated between each adjacent pair of points. This generates a continuous blending curve which has a continuous first derivative at the internal data points.

The equation for the blending curve C(t) is cubic. The blending cubic curve is defined between points P_1 and P_2, but it does not pass through P_0 and P_3. "This behavior makes parabolic blending curve a different interpolation scheme than one which passes a cubic through four points on a curve. Parabolic blending can be used only for internal segments of a curve. Each of the two end segments must be a single parabola, defined through the first and last three data points, respectively" (Rogers and Adams 1976). The formulations of the blending curve $C_i(t_i)$ and the two end parabola segments are easily integrated into the ROBOPATH computer program to generate points on a Cartesian robot path. Actually, these points are Nx1 vectors of position and orientation of end-effector along its path.

Successive points are added, one by one, to determine the continuous cubic blending curve segments. If the end-effector path shape is not correct, the last point or points are deleted and a new path shape is defined using alternate points.

3.2.2 BEZIER CURVES

A Bezier curve is associated with the vertices of a **characteristic polygon** which uniquely define the curve shape. These vertices are also called **control points**. Only the first and last vertex points of the polygon actually lie on the curve; however, the other vertices define the derivatives, order, and shape of the curve. In other words, a general Bezier polynomial curve passes through the points P_0 and P_n, where the tangents are in the directions of the vectors P_0P_1 and $P_{n-1}P_n$. Various Bezier cubic curve segments are shown in Figure 3.2 (Mortenson 1985).

As Faux and Pratt (1985) pointed out, the advantage of higher order Bezier curves is that several orders of continuity can be achieved between segments of compound curves. For example, a fifth-order, or quintic, Bezier polynomial permits the specification of end points, end tangents, and curvature at both ends. Even though the shape of the characteristic polygon still gives some indication of the shape of the associated curve, the relationship becomes weaker as the order of the curve is increased. In general, the rth derivative at an end point must be determined by its r neighboring vertices. This permits us almost unlimited control of the continuity at the joints between curve segments of a composite Bezier curve (Faux and Pratt 1985; Mortenson 1985; Newman and Sproull 1979; Rogers and Adams 1976; Barsky 1984).

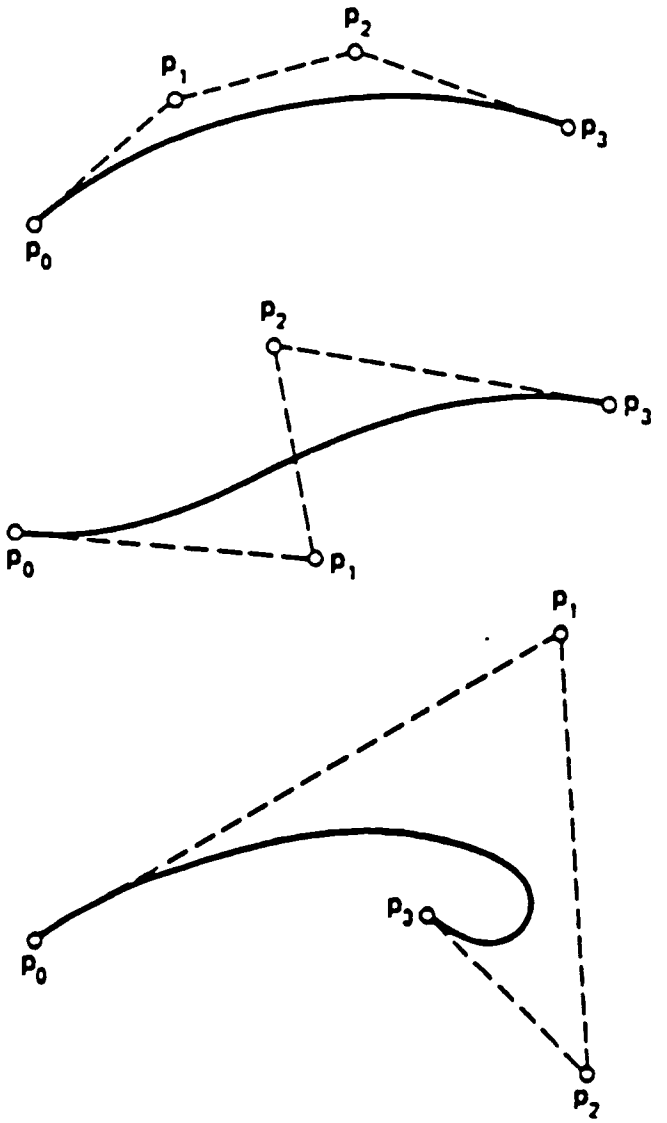


Figure 3.2: Cubic Bezier curves.

The mathematical basis of the Bezier curve is a Bernstein polynomial blending function which interpolates between the first and last vertices. In general, an nth order Bezier polynomial is specified by n+1 vertices. The Bezier curve points are given by

$$P(u) = \sum_{i=0}^{n} P_i B_{i,n}(u) \qquad u\epsilon[0,1] \qquad (3.11)$$

where the vectors P_i represent the n+1 vertices of a characteristic polygon and $B_{i,n}(u)$ is the Bernstein polynomial function

$$B_{i,n}(u) = C(n,i)u^i(1-u)^{n-i} \qquad (3.12)$$

and where C(n,i) is the binomial coefficient

$$C(n,i) = \frac{n!}{i!(n-i)!} \qquad (3.13)$$

Equation (3.11) can be expanded for curves defined by four, and five points to become familiar with the polynomial forms produced.

For four points, n=3, and

$$P(u) = (1-u)^3P_0 + 3u(1-u)^2P_1 + 3u^2(1-u)P_2 + u^3P_3 \qquad (3.14)$$

For five points, n=4, and

$$P(u) = (1-u)^4P_0 + 4u(1-u)^3P_1 + 6u^2(1-u)^2P_2 + 4u^3(1-u)P_3 + u^4P_4 \qquad (3.15)$$

The blending functions are the key to the behavior of Bezier curves. By specifying multiple coincident points at a vertex, the Bezier curve can be pulled in closer and closer to that vertex. The degree of polynomial representing the Bezier curve may be increased without changing the number of sides and shape of the characteristic polygon by adding points coincident with existing vertex points.

P(u) which is a point on the Bezier curve is an nxl vector in n dimensional space. In general, **P**(u) is the 6xl vector, which defines the position and orientation of the robotic manipulator's end-effector on its path, and u is a normalized distance along the robot path. By writing Equation (3.14), the Bezier curve representation of the robot path becomes:

$$\mathbf{P}(u) = [u^3 \; u^2 \; u \; 1] \begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 3 & 0 & 0 \\ 1 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \end{bmatrix} . \qquad (3.16)$$

In a similar manner, Equation (3.15) can be written as

$$\mathbf{P}(u) = [u^4 \; u^3 \; u^2 \; u \; 1] \begin{bmatrix} 1 & -4 & 6 & -4 & 1 \\ -4 & -12 & -12 & 4 & 0 \\ 6 & -12 & 6 & 0 & 0 \\ -4 & 4 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 \end{bmatrix} \begin{bmatrix} \mathbf{P}_0 \\ \mathbf{P}_1 \\ \mathbf{P}_2 \\ \mathbf{P}_3 \\ \mathbf{P}_4 \end{bmatrix} . \qquad (3.17)$$

These equations can be expressed more generally as

$$\mathbf{P}(u) = \mathbf{UM}_n\mathbf{B}_n \qquad (3.18)$$

The subscript n is necessary to distinguish the degree of the Bezier curve intended. $\mathbf{UM}_n$ is the blending-function matrix, and $\mathbf{B}_n$ is a matrix of geometric coefficients.

Obviously, the composition of these matrices varies with the number of vertices (n+1). These matrix representations of the Bezier curves for robot paths are easily integrated into ROBOPATH computer program.

Since Bezier curves are variation-diminishing, they never oscillate wildly away from their defining control points. Bezier curves do not provide localized control. Moving any control point will change

the shape of every part of the curve. Consequently, the location of each control point will influence the curve location almost everywhere.

Bezier curves are special cases of the more general B-spline curves that we consider next.

3.2.3 B-SPLINE CURVES

A B-spline curve, which is an approximation curve technique, is constructed over the complete knot set by splines. The B-spline curve avoids the problem of global propagation of change by using a set of blending functions that has only local influence and depends on only a few neighboring control points. Thus, it is possible to make local modifications to our existing Cartesian robot path without a complete recomputation. This might be a very useful strategy in developing a collision-free robot path in Cartesian space.

B-spline curves are similar to Bezier curves in that a set of blending functions combines the effects of n+1 control points P_i given by

$$P(u) = \sum_{i=0}^{n} P_i N_{i,k}(u) \qquad (3.19)$$

where $P(u)$ is the position vector along the B-spline curve, as a function of the parameter u, and $N_{i,k}(u)$ are the blending functions. For B-spline curves, the degree of these blending function polynomials is controlled by parameter k and is usually independent of the number of control points. The parameter k controls the order and degree of the resulting polynomial in u and also controls the continuity of the curve. The B-spline blending functions are defined by the following recursion formulas:

$$N_{i,1}(u) = \begin{cases} 1 & \text{if } t_i \le u < t_{i+1} \\ 0 & \text{otherwise} \end{cases} \qquad (3.20)$$

and

$$N_{i,k}(u) = \frac{(u - t_i)N_{i,k-1}(u)}{t_{i+k-1} - t_i} + \frac{(t_{i+k} - u)N_{i+1,k-1}(u)}{t_{i+k} - t_{i+1}} \qquad (3.21)$$

where the values of t_i are elements of a knot vector. They relate the parametric variable u to the control points $\mathbf{P}_i$ (Rogers and Adams 1976; Barsky 1984; Mortenson 1985; de Boor 1972).

All the B-splines used in this study are the uniform non-periodic B-splines with order k=4. For a non-periodic open curve, the knot values t_i are chosen with the following rule:

$$\begin{aligned} t_i &= 0 && \text{if } i < k \\ t_i &= i-k+1 && \text{if } k \le i \le n \\ t_i &= n-k+2 && \text{if } i > n \end{aligned} \qquad (3.22)$$

with $0 \le i \le n + k$. The range of the parametric variable u is $0 \le u \le n - k + 2$. Since the denominators in Equation (3.21) can become zero, Equation (3.21) adopts the convention 0/0 = 0.

B-spline curves can be defined with multiply-coincident control points. A duplicate intermediate knot value indicates that a multiple vertex or span of zero length occurs at a point. The curve's shape is pulled in closer to the multiple vertex. This makes it possible to define Cartesian robot paths with sharp corners if it is desired. Each segment of a B-spline curve is influenced by only k control points, and conversely each control point influences only k curve segments. The "interior" blending functions are independent of n, except for the blending functions influenced by the end points or those very near them

depending on parameter k. The only restrictions on the specification of the knot vector are that the same value cannot appear more than k times and that the knots must be in non-decreasing order (Barsky 1984; Mortenson 1985).

There are computational advantages to reparametrizing the interval so that $0 \leq u < 1$ and then identifying the interval by subscripting P(u) as $P_i(u)$ for the ith interval. An expression for B-spline curve points P(u) over an arbitrary segment of the curve (for the interval $i-1 \leq u < i$) can be easily written in matrix notation. P(u) is generally the 6x1 vector for the position and orientation of the manipulator's end-effector on its path. Equation (3.19) can be expanded for the curves defined by parameter k=3 and k=4 to become familiar with the polynomial forms produced.

The analogous form for an open B-spline curve with k=3 is

$$P_i(u) = \frac{1}{2}\,[u^2 \;\; u \;\; 1]\begin{bmatrix} 1 & -2 & 1 \\ -2 & 2 & 0 \\ 1 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \end{bmatrix}$$

$$\text{for } i\epsilon[1{:}n-1] \; . \qquad (3.23)$$

The analogous form for an open cubic B-spline (k=4) is

$$P_i(u) = \frac{1}{6}\,[u^3 \;\; u^2 \;\; u \;\; 1]\begin{bmatrix} -1 & 3 & -3 & 1 \\ 3 & -6 & 3 & 0 \\ -3 & 0 & 3 & 0 \\ 1 & 4 & 1 & 0 \end{bmatrix}\begin{bmatrix} P_{i-1} \\ P_i \\ P_{i+1} \\ P_{i+2} \end{bmatrix}$$

$$\text{for } i\epsilon[1{:}n-2] \; . \qquad (3.24)$$

These equations can be expressed in the form of general formulation for specific k values:

$$P_i(u) = U_k M_k P_k \qquad i\epsilon[1{:}n+2-k] \qquad (3.25)$$

where $U_k = [u^{k-1}\ u^{k-2}\ \ldots\ u\ \ 1]$ and

$P_k = [P_j] \quad j\epsilon[i-1:i+k-2]$ for open curves.

The number of segments i is determined for the open B-spline curves. Matrix representations of the B-spline curves for robot's end-effector paths are easily integrated into ROBOPATH computer program.

Since k=4, the curve is a cubic B-spline, second-order continuity can be expected throughout the curve. A fourth order B-spline curve is a piecewise cubic spline. Thus, a fourth order curve is continuous in first and second derivative, as well as position, along the entire robot path. If the order k equals the number of polygon vertices, and there are no multiple vertices, then a Bezier curve will be generated. The curve produced lies closer to the defining polygon as the order of the curve decreases. When the order of the curve k=2, the generated curve is a series of straight lines which are identical to the defining polygon. As the order of a curve increases, the resulting curve's shape looks less like the defining polygon shape. Therefore, increasing the order tightens the curve. The non-periodic B-spline is used to model open curves in Figure 3.3 (Mortenson 1985).

B-spline curves and Bezier curves have many advantages in common (Newman and Sproull 1979; Mortenson 1985). The main advantage of the B-spline curves over the Bezier curves is the local control of curve shape. This gives the ability to add control points without increasing the degree of the B-spline curve.

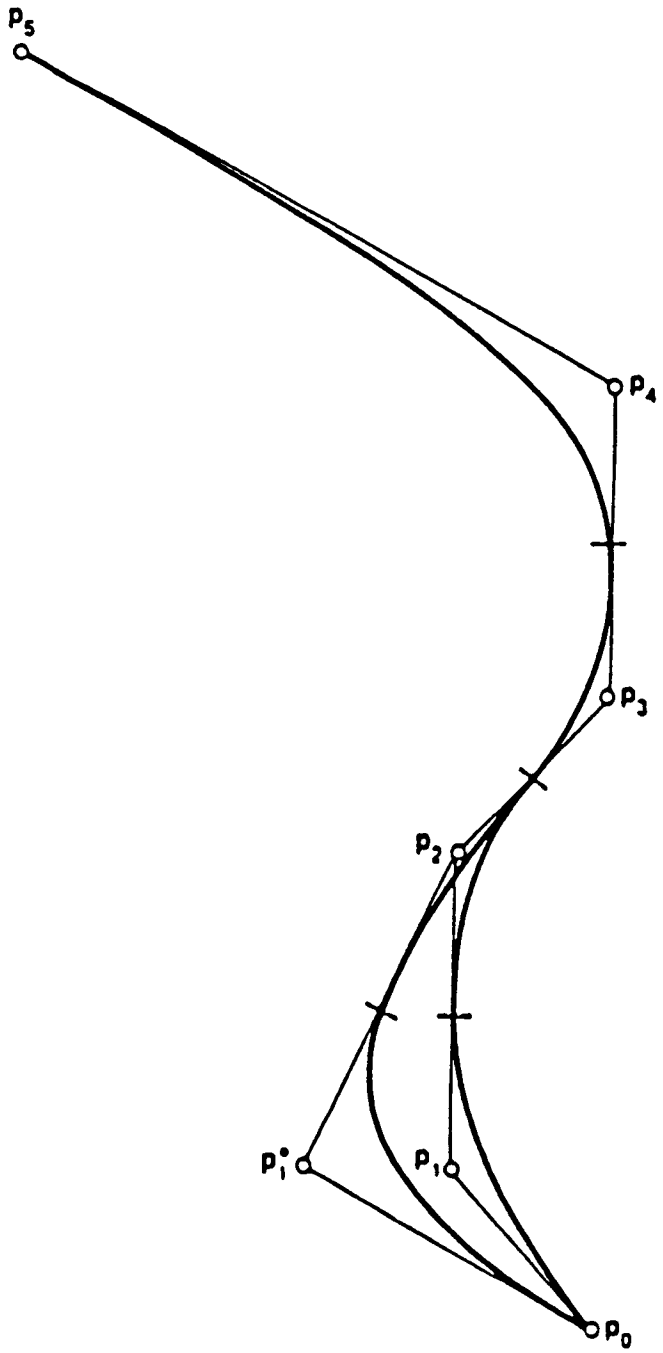


Figure 3.3: Non-periodic B-spline curve: n=5, k=3.

3.3 JOINT TRAJECTORY GENERATION

In this study, to find a joint trajectory that approximates the desired path closely, the Cartesian path points are transformed into N sets of joint displacements, with one set for each joint. Since cubic polynomial trajectories are smooth and have small overshoot of angular displacement between two adjacent knot points, the idea of using cubic spline polynomials is adopted to fit the trajectory segment between two adjacent knot points in joint space. The continuity conditions for joint displacement, velocity, and acceleration must be satisfied on the entire trajectory for the Cartesian robot path. The cubic spline functions are expressed in terms of time intervals between adjacent knot points (Lin et al. 1983; Lin and Chang 1985; Fu et al. 1987).

3.3.1 CUBIC SPLINE JOINT TRAJECTORY

To design the joint trajectories, n Cartesian knot points are first transformed into joint position vectors $(\theta_{11}, \theta_{21}, \ldots, \theta_{N1})$, $(\theta_{12}, \theta_{22}, \ldots, \theta_{N2})$, $\ldots$, $(\theta_{1n}, \theta_{2n}, \ldots \theta_{Nn})$ using the inverse manipulator kinematics routine, where θ_{ji} is the angular displacement of joint j at the ith knot point corresponding to position and orientation of end-effector along its path, P_i which is $P(t_i)$. We now consider the generation of a cubic spline trajectory for each joint j which fits the sequence of joint positions $[\theta_{j1}(t_1), \theta_{j2}(t_2), \ldots, \theta_{jn}(t_n)]$, where $0 = t_1 \leq t_2 \leq \ldots \leq t_n = T$ is a time sequence which indicates when the end-effector should pass through these joint knots (Lin et al. 1983; Lin and Chang 1985; Fu et al. 1987).

As de Boor (1978) pointed out, the smooth curve constituting the path can be characterized by cubic splines. A cubic spline is the

lowest degree approximation to a function obtained when the magnitude and the derivatives of the function at the two end points are specified.

We now consider the generation of a piecewise cubic spline $\phi_{ji}(t)$ for joint j between the Cartesian knot points P_i and P_{i+1}, defined on the time interval $[t_i,t_{i+1}]$. According to the theory of cubic spline functions (Ahlberg et al. 1967; de Boor 1978; Faux and Pratt 1985), the joint position at t_i must equal to θ_{ji}, the joint velocities at t_1 and t_n must equal to zero and the joint velocities and accelerations must be continuous on the time interval $[t_2,t_{n-1}]$ (Lin and Chang 1985).

Faux and Pratt (1985) gives a method which is more usually used in practice gives reasonable accuracy for fairly uniform knot spacings. We initially consider only a piecewise cubic spline function defined in $[t_i,t_{i+1}]$ for joint j, whose time interval is $h_i = t_{i+1} - t_i$, as shown in Figure 3.4.

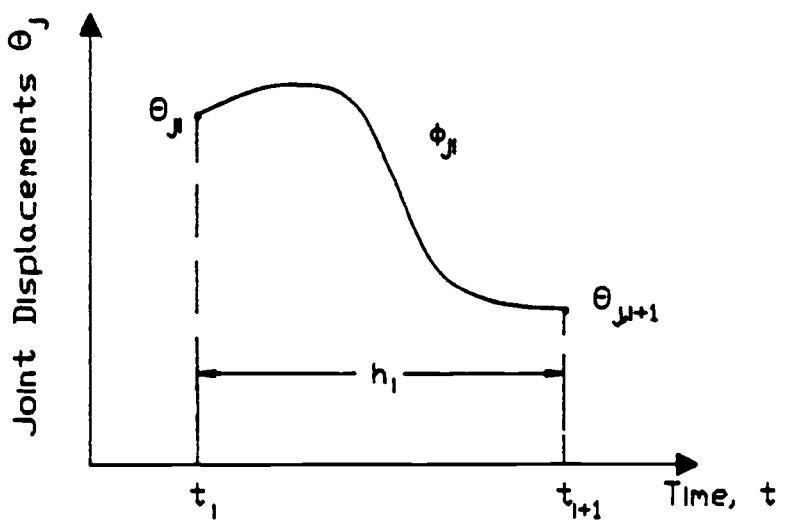


Figure 3.4: A single span of the spline function.

The equations for the unknown joint acceleration $\dot{\omega}_{ji}$ at t_i can be written in matrix form by imposing the conditions mentioned above to solve the problem of joint trajectory interpolation

$$\begin{bmatrix} \dot{\omega}_{j1} \\ \dot{\omega}_{j2} \\ \dot{\omega}_{j3} \\ \vdots \\ \dot{\omega}_{j,n-1} \\ \dot{\omega}_{jn} \end{bmatrix} = \begin{bmatrix} 2h_1 & h_1 & & & 0 \\ h_1 & 2(h_1+h_2) & h & & \\ & h_2 & 2(h_2+h_3) & h_3 & \\ & & & & \\ & & & & \\ 0 & & h_{n-2} & 2(h_{n-2}+h_{n-1}) & h_{n-1} \\ & & & h_{n-1} & 2h_{n-1} \end{bmatrix}^{-1}$$

$$\begin{bmatrix} 6(\theta_{j2} - \theta_{j1})/h_1 \\ 6[(\theta_{j3} - \theta_{j2})/h_2 - (\theta_{j2} - \theta_{j1}/h_1] \\ 6[(\theta_{j4} - \theta_{j3})/h_3 - (\theta_{j3} - \theta_{j2})/h_2] \\ \vdots \\ 6[(\theta_{jn} - \theta_{j,n-1})/h_{n-1} - (\theta_{j,n-1} - \theta_{j,n-2})/h_{n-2}] \\ -6(\theta_{jn} - \theta_{j,n-1})/h_{n-1} \end{bmatrix}$$

for $j = 1,2,\ldots,N$ (3.26)

where $h_i = t_{i+1} - t_i$ is the time spent in traveling segment i and N=2 in our experiment. We have a symmetric tridiagonal system of order n. Therefore, a solution to such a system may be computed accurately and efficiently using well-established TRI technique, as explained in Cheney and Kincaid (1985).

Since the polynomial $\phi_{ji}(t)$ is cubic, its first-time derivative $\phi'_{ji}(t)$ must be a quadratic function of time t and its second-time derivative $\phi''_{ji}(t)$ must be a linear function of time t. Hence, $\phi''_{ji}(t)$ can be expressed at any point on the span by the linear interpolation formula

$$\phi''_{ji}(t) = \frac{(t_{i+1} - t)}{h_i} \dot{\omega}_{ji} + \frac{(t - t_i)}{h_i} \dot{\omega}_{j,i+1}$$

$$\text{for } j = 1,2,\ldots,N \text{ and } i = 1,2,\ldots,n-1. \qquad (3.27)$$

Integrating $\phi''_{ji}(t)$ twice and satisfying the boundary conditions of $\phi_{ji}(t_i) = \theta_{ji}$ and $\phi_{ji}(t_{i+1}) = \theta_{j,i+1}$ leads to the following interpolating functions:

$$\phi_{ji}(t) = \frac{\dot{\omega}_{ji}}{6h_i}(t_{i+1} - t)^3 + \frac{\dot{\omega}_{j,i+1}}{6h_i}(t - t_i)^3 + \left\{\frac{\theta_{j,i+1}}{h_i} - \frac{h_i\dot{\omega}_{j,i+1}}{6}\right\}(t - t_i) + \left\{\frac{\theta_{ji}}{h_i} - \frac{h_i\dot{\omega}_{ji}}{6}\right\}(t_{i+1} - t)$$

$$\text{for } j = 1,2,\ldots,N \text{ and } i = 1,2,\ldots,n-1. \qquad (3.28)$$

Differentiating Equation (3.28) with respect to t, we have

$$\phi'_{ji}(t) = -\frac{\dot{\omega}_{ji}}{2h_i}(t_{i+1} - t)^2 + \frac{\dot{\omega}_{j,i+1}}{2h_i}(t - t_i)^2 + \left\{\frac{\theta_{j,i+1}}{h_i} - \frac{h_i\dot{\omega}_{j,i+1}}{6}\right\} + \left\{\frac{\theta_{ji}}{h_i} - \frac{h_i\dot{\omega}_{ji}}{6}\right\}$$

$$\text{for } j = 1,2,\ldots,N \text{ and } i = 1,2,\ldots,n-1. \qquad (3.29)$$

Imposing the continuity of $\phi'_{ji}(t)$, we must have $\phi'_{ji}(t_i) = \phi'_{j,i+1}(t_i)$ at the interior times t_i $(2 \leq i \leq n-1)$. Differentiating Equation (3.27) with respect to t, we have the jerk which is the rate of change of acceleration:

$$\phi'''_{ji}(t) = \frac{\dot{\omega}_{j,i+1} - \dot{\omega}_{ji}}{h_i}$$

for $j = 1,2,\ldots,N$ and $i = 1,2,\ldots,n-1$. (3.30)

Since the above banded inverse matrix of Equation (3.26) is always nonsingular if the time intervals h_i are positive, the cubic polynomial joint trajectory always has a unique solution. Algorithm for designing the cubic spline joint trajectory with tridiagonal system is integrated into ROBOPATH computer programming by satisfying the continuity of joint position, velocity, and acceleration on the entire time interval $[t_1,t_n]$.

3.4 REMARKS

Trajectory planning is basically done in joint space. End-effector's (or tool's) path is interactively generated in Cartesian space by using polynomial curve techniques which are very effective in CAD. Cubic spline joint polynomials are used to spline n interpolation points generated on the desired end-effector path by the inverse manipulator kinematics. Curve techniques are easily used to program into ROBOPATH computer program. Cubic polynomial joint trajectory technique is an off-line robot motion planning technique. Recent works by Lin et al. (1983) and Lin and Chang (1985) have also demonstrated that an efficient approach to optimum control of robotic manipulators may consist of splitting the problem into two tasks: off-line programming of a robot path along which an optimization can be achieved, followed by an on-line process during which the robotic manipulator is assumed to track the path.

CHAPTER 4

OPTIMIZATION OF POLYNOMIAL JOINT TRAJECTORIES

4.1 INTRODUCTION

An important research effort has been made to develop efficient optimization techniques which could help to solve efficiently specific problems of design in engineering since the 1960's. The purpose of numerical optimization is to provide a computer tool to aid the designers in the design problems. In this study, the idea of nonlinear optimization using the Modified Hooke and Jeeves Direct Search Method is adopted. This optimization technique which was developed in the Department of Mechanical Engineering at Oregon State University (Park 1988) allows us to solve the nonlinear multiple objectives problems by the combination of goal programming and the modified pattern search methods.

Since minimum-time joint trajectory is a constrained nonlinear optimization problem with a single objective function, his optimization algorithm was modified to apply to our optimization problem. Therefore, the optimization method used in this study will be called the Modified Pattern Search Method With Goal Programming which determines the search direction by utilizing n joint knot points in the variable space.

Since the speed of operation affects the productivity of present robotic applications, the total traveling time for the robotic manipulator should be minimized to maximize the speed of operation. Therefore, the optimization problem is to adjust the time intervals subject to the constraints on the joint positions, velocities, accelerations, jerks, torques, and end-effector acceleration within each

of the n-1 pieces of polynomial joint trajectory to minimize the total traveling time. N joint has to be considered simultaneously since the optimization problem is in the joint space coordinates to minimize the robot path for the traveling time. For this problem, we successfully applied the modified pattern search method with goal programming by carefully handling the constraints on the joint trajectories.

The topic of this chapter is to provide a systematic methodology for optimization of polynomial joint trajectories. It is mainly divided into four sections: Introduction, Optimization Using the Modified Pattern Search Method with Goal Programming, Optimization of Polynomial Joint Trajectories, and Remarks.

4.2 OPTIMIZATION USING MODIFIED PATTERN SEARCH METHOD WITH GOAL PROGRAMMING

There are many algorithms for the constrained optimizations. Most of these algorithms are very complicated. However, there is an easy way for the constrained objective optimization which is the modified pattern search method with goal programming. This method has an escape algorithm and a starting point algorithm in addition to the original method. These algorithms are simple and effective. The objective function has a goal which is decided by a decision maker for the optimization.

4.2.1 PATTERN SEARCH METHOD

Hooke and Jeeves (1961) proposed a logically simple strategy of search that made use of prior knowledge and does not require the use of derivatives. Pattern search is a direct search routine for minimizing a

function $f(\underline{x})$ of several variables $\underline{x} = (x_1, x_2, \ldots, x_n)$. The argument $\underline{x}$ is varied until the minimum of $f(\underline{x})$ is obtained. The pattern search routine determines the sequence of values for $\underline{x}$; an independent routine computes the functional values of $f(\underline{x})$. The algorithm consists of two major phases: a "local exploration move" around the base point and a "pattern move (acceleration)" in a direction selected for optimization.

To initiate a local exploration search, $f(\underline{x})$ is evaluated at a base point which is the vector of initial guesses of the independent variables for the first cycle. Then each variable is changed in rotation, one at a time, by incremental amounts, until all the parameters have been so changed. In other words, each time a variable is perturbed and a better value of the economic model is found, this point is used when the next variable is changed rather than returning to the original point (Himmelblau 1972).

Initial values for all the elements of $\underline{x}$ and step size $\Delta\underline{x}$ for independent variables must be provided. The search begins at a base point $\underline{b}_1$. After measuring the criterion at the initial base $\underline{b}_1$, $x_1^{(0)}$ is changed by an amount $+\Delta x_1^{(0)}$, so that $x_1^{(1)} = x_1^{(0)} + \Delta x_1^{(0)}$. If $f(\underline{x})$ is reduced, $x_1^{(0)} + \Delta x_1^{(0)}$ is adopted as the new element in $\underline{x}$. If the increment fails to improve the objective function, $x_1^{(0)}$ is changed by $-\Delta x_1^{(0)}$, and the value of $f(\underline{x})$ again checked as before. If the value of $f(\underline{x})$ is not improved by either $x_1^{(0)} \pm \Delta x_1^{(0)}$, $x_1^{(0)}$ is left unchanged. Then $x_2^{(0)}$ is changed by an amount $\Delta x_2^{(0)}$, and so on, until all the independent variables have been changed to complete one exploratory search. For each step or move in the independent variable, the value of the objective function is compared with the value at the previous point. If the objective function is improved for the given

step, then the new value of the objective function replaces the old one in the testing. However, if a perturbation is a failure, then the old value of $f(\underline{x})$ is retained and the step size $\Delta\underline{x}$ is compared with the tolerance (minimum allowable step size). If the step size is less than the tolerance, then the search will be stopped and the result will be printed for the optimization. Otherwise, the step size will be reduced and another local exploration will be made. When all the independent variables have been perturbed, a new base point $\underline{b}_2$ is established and a "pattern move" takes place.

This pattern move consists of an exploration along a line between the new base point and the previous base point. In other words, the successfully changed variables define a vector that represents a successful direction for minimization. A series of accelerating steps, or pattern moves, is made along this vector as long as $f(\underline{x})$ is decreased by each pattern move. The distance moved beyond the best base point is somewhat larger than the distance between the two base points. Mathematically, this pattern move (extrapolation) is

$$x_{i,0}^{(k+1)} = x_i^{(k+1)} + a(x_i^{(k+1)} - x_i^{(k)}) \qquad (4.1)$$

where $x_{i,0}^{(k+1)}$ becomes a new temporary base point. In this expression i is the variable index, k is the stage index, and a is an acceleration factor that is greater than or equal to 1.0. Once the new temporary base point has been found, an exploration about this point is instituted to see if a better base point can be found. If the temporary base point or any of its neighboring points is a better base, the pattern process repeats using this improved base. Because of the nature of the acceleration factor, each successive pattern extrapolation becomes bolder and bolder until the process oversteps the peak or a ridge. At

this point the previous "best base" is recalled, the local exploration step size is decreased, and the pattern-building process begins again. Once the step size is decreased below a preset tolerance and still no substantial change in the objective function value can be achieved, the search terminates.

4.2.2 ALGORITHM OF THE MODIFIED METHOD WITH GOAL PROGRAMMING

In the pattern search algorithm, only the function values of the interested points are compared. In constrained optimization, constraints are checked first for each test point. If the test point is out of constraints, then local exploration or pattern move is not successful. So, the step size is reduced or another local exploration is tried. If the test point is within constraints, the function value of the test point is compared with the function value of the base point as it is done in unconstrained optimization. In addition, search is done to satisfy the goal for the objective function since the objective function has a goal. In design variable space, the feasible solution area is changed as goal of priority is changed. In other words, goal is variable (moving) constraint having priority.

The above procedure is easy; however, the test point sometimes gets stuck at the boundary of constraints other than the optimum point. Thus, we are unable to get out of this point and the step size is continuously reduced until the step size is less than the tolerance. If this is not the optimum point, an escape algorithm must be applied to get out of this point and continue to search.

4.2.2.1 ESCAPE ALGORITHM

If the base point is on the boundary of constraints and the step size is smaller than the tolerance, there are two possible cases. First, if the base point is better than points around itself, it can be the optimum point. To find the better point, perturbate an escape step along a variable and search until an escape point is found. Then the escape point is compared with the base point. If the escape point is better than the base point, no other escape is done because this algorithm is only to find the approximate acceleration direction. If the escape point is worse than the base point, perturbate along another variable and do the same escape steps as before until all variables are perturbated.

4.2.2.2 STARTING POINT ALGORITHM

It is very difficult and painful to find a good starting point within constraints if there are a lot of variables and complicated constraints. Therefore, an algorithm is needed to find a feasible starting point. When the starting point is beyond constraints, local exploration fails. Thus, the step size is reduced continuously until it is less than the tolerance and search is terminated at the starting point. However, the search can be started out of constraints with little modifications of the algorithm as follows.

When the starting point is out of constraints, inside value is 0. Otherwise, inside value is 1. Search starts with inside = 1. If the starting point is out of constraints, inside value is changed to 0 and continue to search toward the direction minimizing the objective functions without acceleration because acceleration may make it hard to

find the new starting point within constraints. Search is made without considering constraints.

If the optimum point is found beyond constraints, return to the beginning and try another starting point (case 1). If the test point moves from the outside to the inside of constraints, inside value is changed to 1 and continue to search from the feasible search point within constraints (case 2 and 3). In case 3, search is terminated at the constrained optimum point because the unconstrained optimum is not valid.

4.3 OPTIMIZATION OF POLYNOMIAL JOINT TRAJECTORIES

Since the actuator of each joint is subject to saturation and cannot furnish an unlimited amount of torque and force, the total traveling time spent on the specified path approximated by the cubic polynomials is constrained by the maximum values of each joint position, velocity, acceleration, jerk that is the rate of change of acceleration, torque, and end-effector acceleration. The speed of operation affects the productivity for robot applications. In order to maximize the speed of operation or traversing the path, the total traveling time for the robotic manipulator must be minimized. This can be achieved by adjusting the time intervals h_i between two adjacent knot points subject to the joint position, velocity, acceleration, jerk, torque, and end-effector acceleration constraints. N joints must be considered simultaneously to determine a set of optimum values for time intervals $h_1, h_2, \ldots, h_{n-1}$.

The problem can mathematically be expressed as: minimize the objective function subject to the constraints.

$$\text{Minimize: } T = \sum_{i=1}^{n-1} h_i = \sum_{i=1}^{n-1} (t_{i+1} - t_i) \tag{4.2}$$

Subject to:

$$\text{Displacement constraint: } \theta C_{Lj} \le \phi_{ji}(t) \le \theta C_{Uj} \tag{4.3}$$

$$\text{Velocity constraint: } VC_{Lj} \le \phi'_{ji}(t) \le VC_{Uj} \tag{4.4}$$

$$\text{Acceleration constraint: } AC_{Lj} \le \phi''_{ji} \le AC_{Uj} \tag{4.5}$$

$$\text{Jerk constraint: } JC_{Lj} \le \phi'''_{ji} \le JC_{Uj} \tag{4.6}$$

$$\text{Torque constraint: } \tau C_{Lj} \le \tau_j(\theta_{ji}(t), \theta'_{ji}(t), \theta''_{ji}(t)) \le \tau C_{Ui} \tag{4.7}$$

$$\text{End-effector acceleration constraint: } \dot{\vartheta}_{ei}(t) \le EAC \tag{4.8}$$

for $j = 1,2,\ldots,N$ and $i = 1,2,\ldots,n-1$

where T is the total traveling time, L is the lower bound index, U is the upper bound index, θC_j, VC_j, AC_j, JC_j and τC_j are the position, velocity, acceleration, jerk, and torque limits of joint j, and EAC is the limit of end-effector acceleration.

The above constraints are expressed in explicit forms in Chapter 3. Let us summarize them as follows:

Position Constraints:

$$\theta C_{Lj} \le \phi_{ji}(t) = \frac{\dot{\omega}_{ji}}{6h_i}(t_{i+1} - t)^3 + \frac{\dot{\omega}_{j,i+1}}{6h_i}(t - t_i)^3 + \left\{\frac{\theta_{j,i+1}}{h_i} - \frac{h_i\dot{\omega}_{j,i+1}}{6}\right\}(t - t_i) + \left\{\frac{\theta_{ji}}{h_i} - \frac{h_i\dot{\omega}_{ji}}{6}\right\}(t_{i+1} - t) \le \theta C_{Uj} \tag{4.9}$$

where $\dot{\omega}_{ji}$ is the acceleration at joint knot points and equal to $\phi''_{ji}(t)$ if the time instant at which $\phi_{ji}(t)$ passes through the joint knot points is t_i.

Velocity Constraints:

$$VC_{Lj} \le \phi'_{ji}(t) = -\frac{\dot{\omega}_{ji}}{2h_i}(t_{i+1} - t)^2 + \frac{\dot{\omega}_{j,i+1}}{2h_i}(t - t_i)^2 + \left\{\frac{\theta_{j,i+1}}{h_i} - \frac{h_i\dot{\omega}_{j,i+1}}{6}\right\} + \left\{\frac{\theta_{ji}}{h_i} - \frac{h_i\dot{\omega}_{ji}}{6}\right\} \le VC_{Uj} \tag{4.10}$$

The maximum absolute value of velocity exists at t_i, t_{i+1}, or $\bar{t}_i$, where $\bar{t}_i \epsilon [t_i, t_{i+1}]$ and satisfies $\phi''_{ji}(\bar{t}_i) = 0$ (Lin et al. 1983). The velocity constraints then become

$$\max_{t\epsilon[t_i,t_{i+1}]} |\phi'_{ji}| = \max \{|\phi'_{ji}(t_i)|, |\phi'_{ji}(t_{i+1})|, |\phi'_{ji}(\bar{t}_i)|\} \le VC_{Uj}$$

$$\text{for } i = 1,2,\ldots,n-1;\ j = 1,2,\ldots,N \tag{4.11}$$

where

$$|\phi'_{ji}(t_i)| = \left| -\frac{\dot{\omega}_{ji}}{2} h_i + \frac{\theta_{j,i+1} - \theta_{ji}}{h_i} + \frac{(\dot{\omega}_{ji} - \dot{\omega}_{j,i+1})}{6} h_i \right|$$

$$|\phi'_{ji}(t_{i+1})| = \left| \frac{\dot{\omega}_{j,i+1}}{2} h_i + \frac{\theta_{j,i+1} - \theta_{ji}}{h_i} + \frac{(\dot{\omega}_{ji} - \dot{\omega}_{j,i+1})}{6} h_i \right|$$

and

$$|\phi'_{ji}(\bar{t}_i)| = \begin{cases} \left| \dfrac{\dot{\omega}_{ji}\,\dot{\omega}_{j,i+1}}{2(\dot{\omega}_{ji} - \dot{\omega}_{j,i+1})} h_i + \dfrac{\theta_{j,i+1} - \theta_{ji}}{h_i} + \dfrac{(\dot{\omega}_{ji} - \dot{\omega}_{j,i+1})}{6} h_i \right|, \\ \qquad \text{if } \dot{\omega}_{ji} = \dot{\omega}_{j,i+1} \text{ and } t_i \epsilon [t_i, t_{i+1}] \\ 0, \qquad \text{if } \dot{\omega}_{ji} = \dot{\omega}_{j,i+1} \text{ and } t_i \epsilon [t_i, t_{i+1}]. \end{cases}$$

Acceleration Constraints:

$$AC_{Lj} \leq \phi''_{ji}(t) = \frac{(t_{i+1} - t)}{h_i}\dot{\omega}_{ji} + \frac{(t - t_i)}{h_i}\dot{\omega}_{j,i+1} \leq AC_{Uj}$$

$$\text{for } i = 1,2,\ldots,n-1; \quad j = 1,2,\ldots,N . \tag{4.12}$$

The acceleration is a linear function of time between two adjacent knot points. Therefore, the maximum absolute value of acceleration occurs at either t_i or t_{i+1} and equals the maximum of $\{|\dot{\omega}_{ji}|, |\dot{\omega}_{j,i+1}|\}$. Consequently, the acceleration constraints become

$$\max \{|\dot{\omega}_{j1}|, |\dot{\omega}_{j2}|, \ldots, |\dot{\omega}_{jn}|\}, \leq AC_j$$

$$j = 1,2,\ldots,N . \tag{4.13}$$

Jerk Constraints:

$$JC_{Lj} < \phi'''_{ji}(t) = \frac{\dot{\omega}_{j,i+1} - \dot{\omega}_{ji}}{h_i} \leq JC_{Uj}$$

$$\text{for } i = 1,2,\ldots,n-1; \quad j = 1,2,\ldots,N . \tag{4.14}$$

The jerk is the rate of change of acceleration.

Torque Constraints:

The torque $\tau(t)$ can be computed from the dynamic equations of motion [Equ. (2.14)]

$$\tau C_{Lj} \leq \tau_j(t) = M(\phi_{ji}(t))\phi''_{ji}(t) + V(\phi_{ji}(t),\phi'_{ji}(t)) + G(\phi_{ji}(t)) + F(\phi_{ji}(t),\phi'_{ji}(t)) \leq \tau C_{Uj}$$

$$\text{for } i = 1,2,\ldots,n-1; \quad j = 1,2,\ldots,N \tag{4.15}$$

where M, V, G and F are dynamic coefficients depending on the robot arm structure and joint positions and velocities. The torque constraint must also be checked along the path.

End-effector Acceleration Constraint:

$$\dot{v}_{ei}(t) \leq EAC \qquad \text{for } i = 1,2,\ldots,n-1 . \tag{4.16}$$

The acceleration of the end-effector (or the tip) of the manipulator $\dot{\vartheta}_{ei}(t)$ can be computed by using Equ. (2.7). End-effector acceleration with respect to the nonmoving base frame will be calculated by rotating them with the rotation matrix ${}^{0}_{N}R$.

In addition to the above constraints, the lower bound of the vector of time intervals is estimated as (Lin et al. 1983)

$$\underline{h}' \triangleq (h'_1, h'_2, \ldots, h'_{n-1}) =$$

$$= \max_j \left\{ \frac{|\theta_{j2} - \theta_{j1}|}{VC_{Uj}}, \max_j \frac{|\theta_{j3} - \theta_{j2}|}{VC_{Uj}}, \max_j \frac{|\theta_{j,n} - \theta_{j,n-1}|}{VC_{Uj}} \right\}$$

for $j = 1,2,\ldots,N$. (4.17)

With this formulation, the objective is to use modified pattern search method with goal programming that will minimize the total traveling time subject to the joint position, velocity, acceleration, jerk, torque, and end-effector acceleration constraints.

Initial values for all the elements of $\underline{h}$ is defined as the vector of time intervals between selected knots, i.e., $[h_1, h_2, \ldots, h_{n-1}]$. The objective function for $\underline{h}$ is represented by $T(\underline{h})$ and equals $(h_1+h_2+\ldots+h_{n-1})$. Therefore, the optimum path problem is to minimize the objective function $T(\underline{h})$ subject to above constraints. As defined previously, there are totally n-1 variables in this problem.

Initial values of time intervals $h_1, h_2, \ldots, h_{n-1}$ are first guessed; thus, the joint acceleration $\dot{\omega}_{ji}$ will be uniquely determined. Since this is a constrained optimization, above constraints are checked for each initial value of time interval. If the initial time intervals

$(h_1,h_2,\ldots,h_{n-1})$ are out of constraints, then local exploration or pattern move is not successful. Hence, the step size will be reduced or another local exploration will be tried. If the time intervals $(h_1,h_2,\ldots,h_{n-1})$ are within constraints, the function value, $T(\underline{h})$, of the test point is evaluated for the first cycle and then compared with the function value of the base point. If $T(\underline{h})$ is reduced, then the goal for the objective function must be satisfied. Whenever a functional improvement is obtained by satisfying the constraints, a new temporary base point is established. Once this local exploration is complete, a new base point is established and a pattern move takes place. By moving a distance beyond the best base point, a new temporary base point is established. The constraints are then checked at the new temporary base point. If it is out of constraints, then pattern move is not successful. So, the step size is reduced or another local exploration is tried. If the new temporary base point is within constraints, $T(\underline{h})$ value for this base point is compared with the function value of the previous base point. If $T(\underline{h})$ is reduced and the goal for the objective function is satisfied, then the temporary base point is a better base point. Once the new temporary base point has been found, an exploration about this point is instituted to see if a better base point can be found. As it is stated in Section 4.2, if the temporary base point or any of its neighboring points is a better base point, the pattern process repeats using this improved base point. Each successive pattern extrapolation becomes bolder and bolder until the process oversteps the peak or a ridge. At this point the previous best base point is recalled, the local exploration step size is decreased, and the pattern-building process begins again. Once the step size is decreased below a

predetermined tolerance value and still no substantial change in $T(\underline{h})$ value can be achieved, or constraints are violated, the optimization procedure terminates.

If test point gets stuck at the boundary of constraints other than the optimum point during optimization process, then escape algorithm is applied to get out of this test point.

4.4 REMARKS

Any starting point within constraints finds the optimum value under the same goal. The optimum point also depends on the decision of the decision maker.

Of course, the starting point algorithm does not guarantee the feasible search point, but the chance of trial and error for finding the feasible starting point is considerably reduced.

CHAPTER 5

ROBOPATH

5.1 INTRODUCTION

As part of this study, a computer program for the SCARA robot has been written in Turbo Pascal (Version 3.01) on IBM and its compatibles to implement our algorithm for optimum robot paths. This computer program called ROBOPATH that implements the procedure for constructing the specified end-effector path in Cartesian space, the procedure for constructing the cubic polynomial joint trajectory, and the algorithm for minimizing the total traveling time subject to constraints on the joint position, velocity, acceleration, jerk, torque, and end-effector acceleration. For illustration, we consider the first two degrees of freedom of a 4-link SCARA robot with four degrees of freedom. Since we only consider the horizontal link motion of SCARA robot in xy-plane, ROBOPATH implements it as a two degree-of-freedom planar manipulator with revolute joints. ROBOPATH considers the two dimensional paths and end-effector motions with polynomial curves. The inputs to the program include information on the path, manipulator's kinematic and dynamic parameters, constraints on the trajectory and motion of manipulator, and initial guess values for time intervals. The output of the program includes time optimal joint positions, velocities, accelerations, jerks, torques and end-effector acceleration along the path, and the minimum total traveling time for a specified path. The main flow diagram of the ROBOPATH is shown in Figure 5.1.

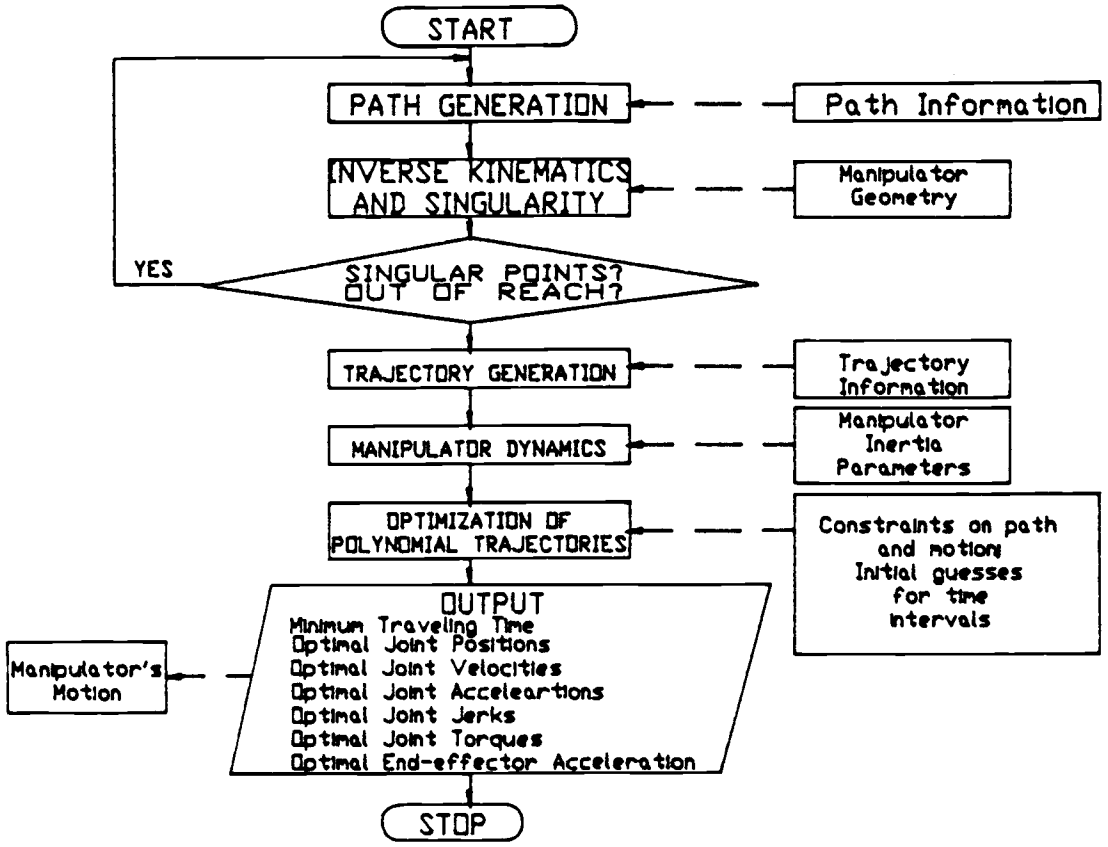


Figure 5.1: Flow diagram of ROBOPATH.

The main parts of ROBOPATH are:

1. End-effector path generation,
2. Inverse kinematics and singularity computation,
3. Cubic polynomial joint trajectory generation,
4. Manipulator dynamics computation,
5. Optimization of polynomial trajectory.

The first four parts compute the information required by the algorithm for the optimization.

The program, ROBOPATH, is an effective tool for improving manipulator dynamic performance, manipulator design, and the design of their tasks and workspaces.

5.2 END-EFFECTOR PATH GENERATION

The end-effector path is generated in the including file PATHGEN which has three options for constructing the path. These options are Bezier, B-spline, and parabolic blending curves. Each curve technique which has different characteristics is previously defined in Section 3.2. The end-effector path is generated from a given set of Cartesian points by selecting the desired curve technique from the end-effector path selection menu. In fact, each curve technique is generating some Cartesian knot points on the path which will be transformed into joint knot points by applying inverse manipulator kinematics to construct the cubic polynomial joint trajectory for each joint.

5.3 INVERSE KINEMATICS AND SINGULARITY COMPUTATION

Inverse manipulator kinematics and singularity computation are done in the including file called KINMATIC, using the method derived in Chapter 2 and Appendix B. The generated Cartesian knot points on the end-effector path using PATHGEN algorithm are input to the KINMATIC algorithm. These Cartesian knot points are transformed into N sets of joint displacements, with one set for each joint. Then each joint displacement is checked for out of reach points and singularity on the path. In case of singularity or out of reach points on the path, the execution of program is aborted and the program, ROBOPATH, prompts the critical point at which case of singularity or out of reach happened. Then the program returns to the end-effector path generation menu to generate a new end-effector path.

In case of multiple solution, we may choose the "elbow-up" or the "elbow-down" solution. In determining θ_2 we have used one of the recurring methods for solving the type of kinematic relationships (Craig 1986). Determine both the sine and cosine of the desired joint angle, and then apply the two argument arctangent. This insures that we have found all solutions. In KINMATIC algorithm, two solutions are called as "near", the nearest solution, and "far", the second solution. We choose the nearest solution in inverse manipulator kinematics; thus, our algorithm can be called economical by minimizing mechanical energy in the motion of manipulator.

5.4 CUBIC POLYNOMIAL JOINT TRAJECTORY GENERATION

The generation of a cubic spline trajectory is done for each joint j which fits the sequence of joint positions $[\theta_{j1}(t_1),\theta_{j2}(t_2),\ldots,$

$\theta_{jn}(t_n)]$, where $0 \leq t_1 \leq t_2 \leq \ldots \leq t_n = T$ is a time sequence which indicates when the end-effector must pass through these joint knot points. Formulation of cubic spline joint trajectory is given in Section 3.3. As it is previously mentioned, piecewise cubic polynomial functions are used to fit the sequence of joint displacements for each of the N joints. Joint positions, velocities, accelerations, and jerks are first computed using the Equations (3.27) through (3.30).

The unknown joint acceleration $\dot{\omega}_{ji}$ at t_i is written in a matrix form, Equation (3.26), which is a symmetric tridiagonal system of order n. A solution for the system is computed accurately and efficiently using the TRI technique in subroutine COMPUTE_JOINT_ACCELERATIONS.

5.5 MANIPULATOR DYNAMICS COMPUTATION

The dynamics of the manipulator are computed using the method derived in Chapter 2. The vector parameters $M(\phi)$, $B(\phi)$, $C(\phi)$, and $G(\phi)$ of Equation (2.15) are computed in an example of closed form dynamic equations for the 2-link manipulator with two degree-of-freedom which is given in Chapter 2. The equations of motion for the manipulator are calculated in ROBOPATH using Equation (2.15). Thus, the joint torques are obtained for desired motion of manipulator.

5.6 OPTIMIZATION OF POLYNOMIAL TRAJECTORY

Optimization of piecewise cubic polynomial trajectory for each joint j is implemented in a subroutine called HJ_OPTIMIZATION. The optimization algorithm has been described in Chapter 4. The constraints on joint position, velocity, acceleration, jerk, torque, and end-effector acceleration along the path are specified in a data file called

SCARA.DAT. Thus, ROBOPATH may be applied to any two degrees of freedom manipulator since manipulator's specifications and constraints are given in an external data file which is created by the user for desired manipulator. The data file can be created in ASCII code using any kind of word processing software. The format of the data file is shown as follows:

L_1, L_2

L_{C1}, L_{C2}

m_{L1}, m_{L2}, m_{L3}

θ_{1min}, θ_{1max}, θ_{2min}, θ_{2max}

θ'_{1min}, θ'_{1max}, θ'_{2min}, θ'_{2max}

θ''_{1min}, θ''_{1max}, θ''_{2min}, θ''_{2max}

θ'''_{1min}, θ'''_{1max}, θ'''_{2min}, θ'''_{2max}

τ_{1min}, τ_{1max}, τ_{2min}, τ_{2max}

a_e

The units are specified as follows:

Link lengths: L = meter

Location of center of masses: L_C = meter

Linkage masses: m_L = kg

Joint constraints:

Position: θ = degree

Velocity: θ' deg/sec

Acceleration: θ'' = deg/sec^2

Jerk: θ''' = deg/sec^3

Torque: τ = Newton-meter

End-effector acceleration: a_e = meter/sec^2

The logic diagram for optimization of piecewise cubic polynomial trajectory is shown in Figure 5.2. For more detail of logic diagram, refer to Park (1988).

5.7 OUTPUT OF ROBOPATH

The output of ROBOPATH can be obtained for the desired robot operation and are graphically plotted on a plotter using graphic softwares. The output of the program basically includes the following along the path:

1. Minimum total traveling time,
2. Time optimal joint trajectories,
3. Time optimal joint velocities,
4. Time optimal joint accelerations,
5. Time optimal joint jerks,
6. Time optimal joint torques, and
7. Time optimal end-effector acceleration.

Examples of the results obtained by ROBOPATH and the various outputs, listed above, are shown in Chapter 6.

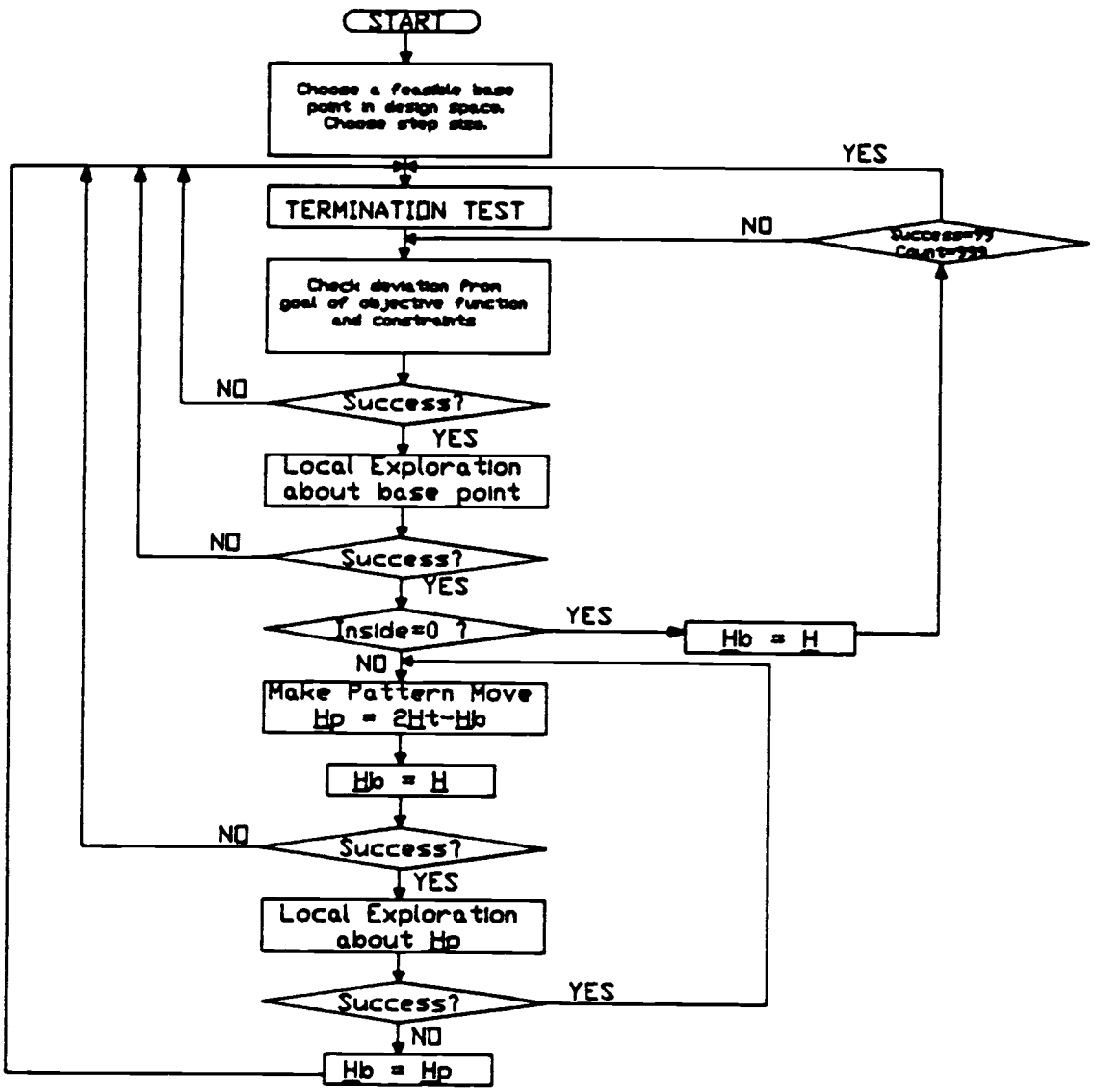


Figure 5.2: Logic diagram for optimization method.

5.8 REMARKS

The program, ROBOPATH, consists of about 2300 lines of code divided into subroutines and including files dealing with different aspects of simulation. Due to memory limitations on the IBM AT and Turbo Pascal Version 3.01, the memory requirements for the program are thus kept within allowable user limits.

Some explanatory instructions are given at the beginning of the program about ROBOPATH. Also, an execution file of ROBOPATH has been created to run it independently from Turbo Pascal.

CHAPTER 6

EXAMPLES AND RESULTS

6.1 INTRODUCTION

Most SCARA robots have two degrees of freedom for the end-effector in addition to the two degrees of freedom for the links considered so far. If the two end-effector coordinates do not change during the motion of the manipulator, the method developed in Chapter 2 may be used. Of course to do this, the mass and inertia of the end-effector and third link would have to be lumped into equivalent properties for a rigid second link (Bobrow 1982). However, in some instances, we may want to specify the position and orientation of the end-effector throughout the motion. For example, a peg may need to be picked up at a part feeder and moved to a hole without tilting for an insertion task. It is desirable to accomplish this motion in minimum-time.

The following examples, obtained by using the computer program ROBOPATH, show the optimal joint trajectories of a two degree-of-freedom planar manipulator on typical end-effector paths in the horizontal xy-plane. The manipulator in the examples is of the type shown in Figure 2.1, with the parameters given in Table 1. The examples demonstrate the practicality of the algorithm and of ROBOPATH for two DOF manipulators and for different shapes of path in their workspace. In addition, the examples show the effect of the shape of the end-effector path, the different path methods, and the actuator limits on the optimal joint trajectories. ROBOPATH can be interactively used to improve manipulator performance, manipulator design, and the design of manipulator work cells by iterating on the manipulator parameters.

TABLE 1. Manipulator's Parameters.

	L (meter)	L_c (meter)	Mass (m) (kg)	Moment of Inertias ($kg\text{-}m^2$)			
				I_{xx}	I_{yy}	I_{zz}	$I_{xy}=I_{xz}=I_{yz}$
Link 1	0.5	0.25	3.38	0.0039	0.0011	0.0039	0.0
Link 2	0.5	0.25	3.38	0.0039	0.0011	0.0039	0.0

6.2 NUMERICAL EXAMPLES

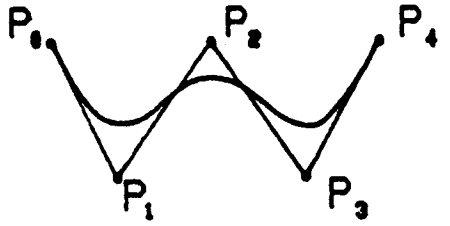
In this section, we considered two types of examples. The first example considered was that of finding how end-effector paths affect the total traveling time and how consistently curve generation methods for end-effector path affect the total traveling time of 2 DOF manipulators. In this type of example, we considered six different end-effector paths generated by using three curve generation methods for three sets of initial time estimates. The second example considered was that of finding how consistently initial time estimates affect the total traveling time of 2 DOF manipulators. For this example, we considered two end-effector paths generated by using three curve generation methods for different sets of initial time estimates.

6.2.1 EXAMPLE 1

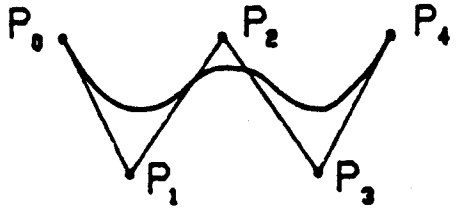
In this type of example, we will apply three proposed curve techniques to design Cartesian end-effector paths of the manipulator with two links and two degrees of freedom as shown in Figure 2.1. The problem is two-fold: First is to find the effects of the end-effector paths on the total traveling time; second is to find how consistently the curve techniques affect the total traveling time.

We applied three curve techniques for six different shapes of end-effector paths in three sets of initial time intervals $h_1, h_2, \ldots, h_{n-1}$. Figure 6.1 shows one of the six end-effector paths in two dimensional view. The other end-effector paths are shown in Appendix C. The Cartesian end-effector path is defined by five control points P_0 (starting point) through P_4 (final goal point). Referring to Figure 6.2, the manipulator tip starts at rest at point $P_0(-0.8,0.5)$, moves along the curve toward points $P_1(-0.45,0.1)$, $P_2(0.1,0.8)$, and $P_3(0.5,0.3)$. Then it moves to the final goal point $P_4(0.8,0.55)$ and stops at P_4. All coordinate units are in meters. Such a path might be required for material handling or assembly applications.

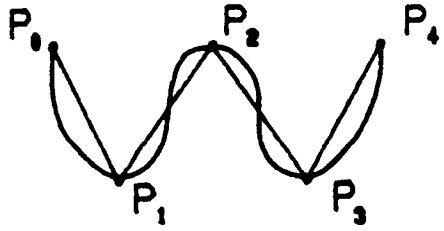
For the given Cartesian control points, each curve technique generates a different end-effector path as shown in Figure 6.1. Five control points are defined to generate a Cartesian path of the end-effector using curve techniques. For these five control points, each curve technique generated some Cartesian knots along the end-effector path as shown in Table 2. Therefore, the manipulator motion is specified by the sequence of these Cartesian knots, i.e. positions and orientations of the end-effector. By means of inverse manipulator kinematics and Jacobian, these Cartesian knots are transformed into sets of joint displacements, with one set for each joint as shown in Table 3.



(a) Bezier Curve



(b) B-spline Curve



(c) Parabolic Blending Curve

Figure 6.1: Cartesian end-effector paths.

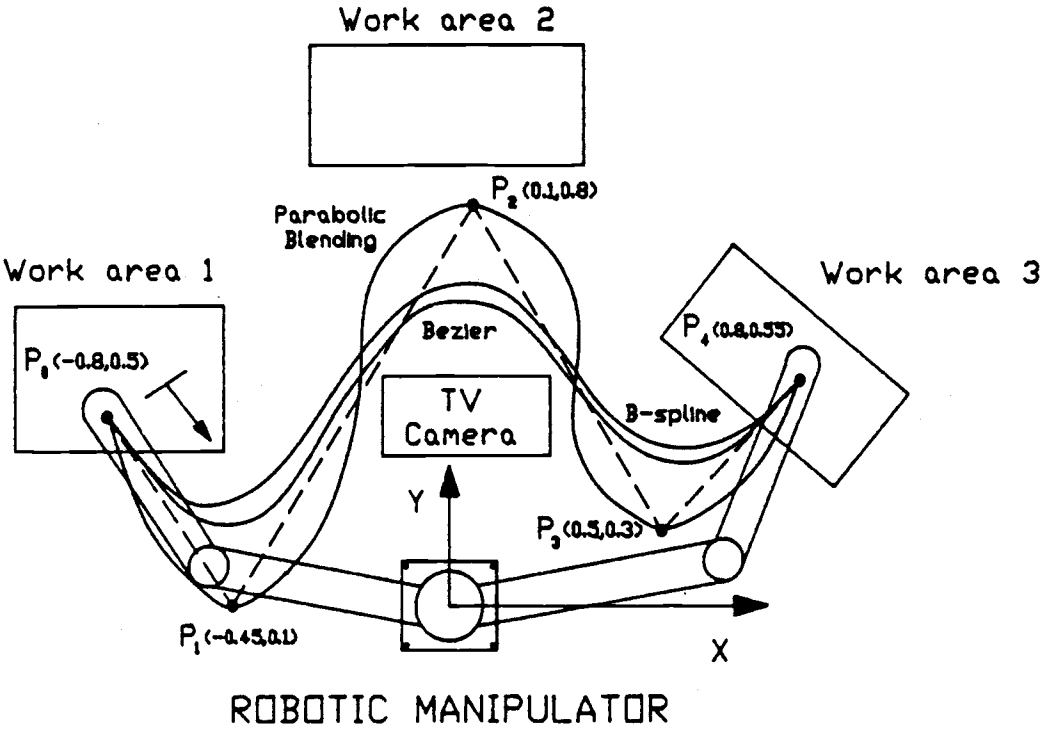


Figure 6.2: The top view of a 2 DOF manipulator on its intuitive path (Example 1).

TABLE 2. Cartesian knots along end-effector path (in meters).

Curves / Knots \ Coor.	Bezier		B-spline (k=4)		Parabolic blending	
	X	Y	X	Y	X	Y
1	-0.800	0.500	-0.800	0.500	-0.800	0.500
2	-0.609	0.386	-0.551	0.324	-0.614	0.268
3	-0.395	0.385	-0.327	0.331	-0.450	0.100
4	-0.172	0.427	-0.123	0.423	-0.035	0.030
5	0.050	0.466	0.063	0.500	0.100	0.800
6	0.263	0.481	0.236	0.494	0.288	0.609
7	0.461	0.476	0.408	0.450	0.500	0.300
8	0.640	0.481	0.592	0.444	0.640	0.396
9	0.800	0.550	0.800	0.550	0.800	0.550

TABLE 3. Joint displacements for Cartesian knots (in degrees).

Curves / Knots \ Joint	Bezier		B-spline (k=4)		Parabolic blending	
	Joint 1	Joint 2	Joint 1	Joint 2	Joint 1	Joint 2
1	128.62	38.74	128.62	38.74	128.62	38.74
2	103.75	87.72	99.26	100.52	108.52	95.84
3	79.24	112.98	72.31	124.56	104.92	125.10
4	49.37	125.17	42.38	127.76	52.22	174.75
5	21.80	124.15	23.13	119.48	46.60	72.54
6	4.52	113.57	7.65	113.66	17.07	95.31
7	-2.58	97.01	-4.79	105.21	-23.37	108.66
8	0.14	73.56	-5.44	84.62	-9.43	82.40
9	20.63	27.75	20.63	27.75	20.63	27.75

The manipulator motion takes place in the horizontal plane. Thus, this is a planar problem which uses the full nonlinear manipulator equations of motion with the inertial properties given in Table 1. The manipulator is initially at rest and comes to a full stop at the end of the last minimum time interval. Hence, the velocities at t_1 and t_n must be zero, $\phi'(t_1) = \phi'(t_n) = 0$, although the corresponding acceleration and jerk may have some finite values. The constraints on the joint

positions, velocities, accelerations, jerks, and torques are given in Table 4. The constraint on the end-effector acceleration is 2g, where g is the gravity acceleration 9.81 m/sec^2. For simplifying calculations, suppose that the characteristics of actuators and the frictional effects are negligible. The actuator torque limits were chosen to be constant although they are actually nonlinear functions of the joint angles. The main reason for using constant torque limits is that it is easier to interpret the numerical output.

TABLE 4. Joint constraints for optimization.

Joints / Constraints	Joint 1		Joint 2	
	Lower	Upper	Lower	Upper
Position (degrees)	-240	240	-270	270
Velocity (degrees/sec)	-200	200	-150	150
Acceleration (degrees/sec^2)	-90	90	-70	70
Jerk (degrees/sec^3)	-60	60	-60	60
Torque (Newton-m)	-20	20	-10	10

The optimization algorithm computes the lower bound of the vector of the time interval using Equ. (4.17) for Bezier path

$$\underline{h}_{min} = [\ 0.33 \quad 0.17 \quad 0.15 \quad 0.14 \quad 0.09 \quad 0.11 \quad 0.16 \quad 0.31\ ] .$$

In this example, three sets of initially guessed time intervals are chosen and they are 3, 5, and 7 seconds for each joint trajectory.

Furthermore, the three curve techniques for end-effector path are Bezier, B-spline, and parabolic blending curves. The optimization results of these three curve techniques are presented in Table 5.

TABLE 5. The results of optimization for total traveling time.

Curve Techniques / Time Intervals ($\underline{h}$) (sec)	Total Traveling Time			
	Initial Guess	Bezier Curve	B-spline Curve (k=4)	Parabolic Blending Curve
3	24	6.288	7.267	13.074
5	40	6.389	7.367	12.746
7	56	6.105	7.252	12.743

According to the above table, the minimum total traveling time is obtained using the set of 7 seconds for the initial time intervals, and the shortest path in minimum time is given by the Bezier curve technique. The results of the optimization show that different locations of a path and different path's shape of the end-effector along the path resulted in different total traveling times.

For this example, the optimization algorithm performs 13, 11 and 10 cycles of search sequences to obtain the final solution for Bezier, B-spline (k=4) and parabolic blending curves, respectively. The results are given in Table 6. The total traveling time is reduced in each cycle. All h_i's are adjusted simultaneously to achieve a lower total traveling time. Thus, the time intervals $h_1, h_2, \ldots, h_{n-1}$ are highly related. This improvement in the traveling time required no change in the manipulator's hardwares.

TABLE 6. The results of optimum time interval values.

Time Intervals ($\underline{h}$)	Initial Guesses (sec)	Optimum Values (sec)		
		Bezier	B-spline	Parabolic Blending
h_1	7	1.495	1.749	1.762
h_2	7	0.618	0.670	0.943
h_3	7	0.618	0.768	2.199
h_4	7	0.568	0.625	2.321
h_5	7	0.475	0.507	1.493
h_6	7	0.439	0.577	1.359
h_7	7	0.493	0.647	0.994
h_8	7	1.400	1.708	1.671
Total Time (sec)	56	6.105	7.252	12.743

Figures 6.3 through Figure 6.22 show graphic results of the optimum joint positions, velocities, accelerations, jerks, torques, and end-effector accelerations for this example. They are the desired optimum values that the manipulator must follow to achieve a minimum traveling time.

Finally, we can compare all six different end-effector paths (given in Appendix C) by taking the total traveling time of the Bezier curve technique as a base reference which will be 1. Thus, we can normalize the total traveling time for other curve techniques, B-spline and parabolic blending curves. The comparison of curve techniques for the total traveling time is shown in Table 7, normalizing the total traveling time.

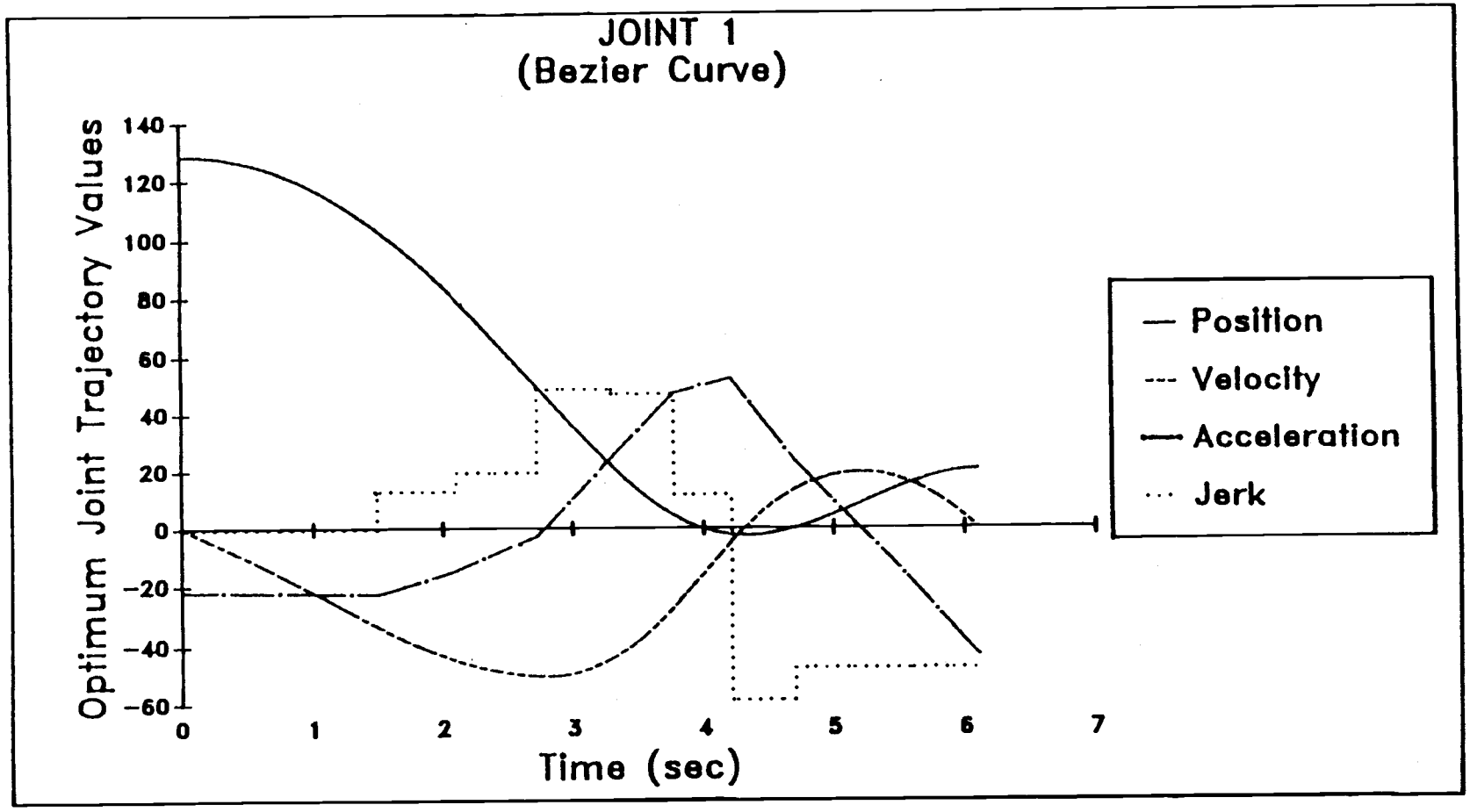


Figure 6.3: Optimum joint trajectories of Bezier path for joint 1.

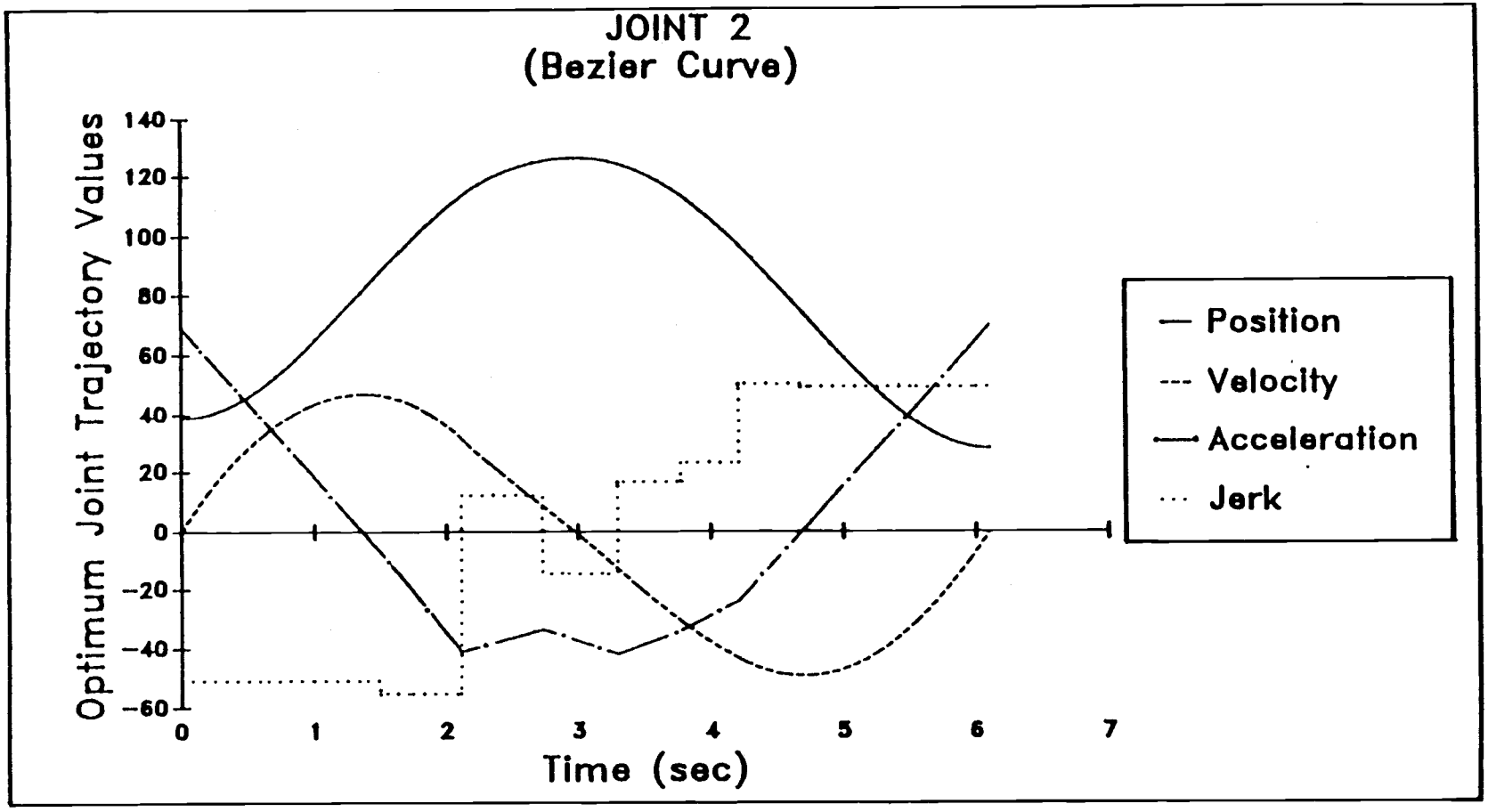


Figure 6.4: Optimum joint trajectories of Bezier path for joint 2.

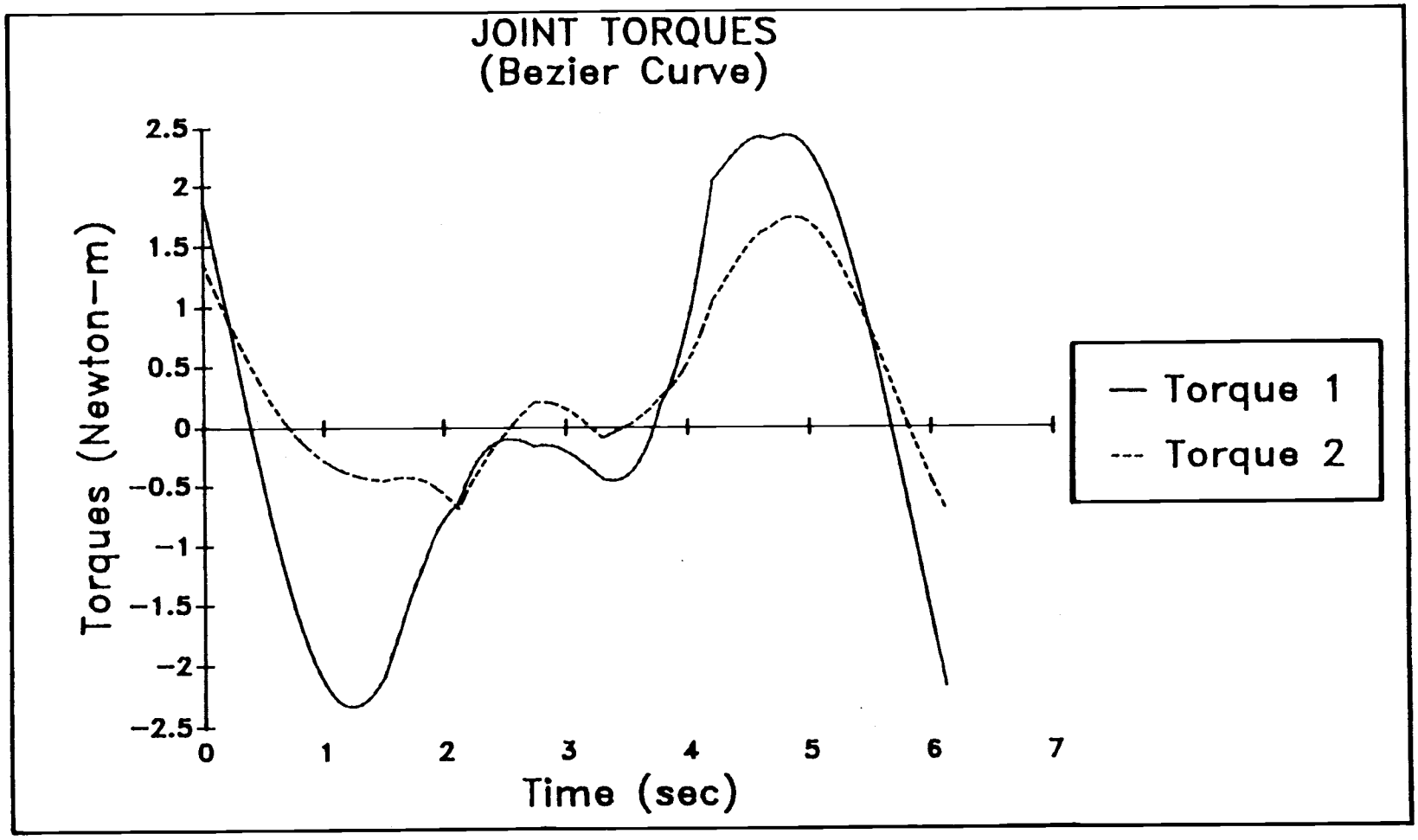


Figure 6.5: Optimum joint torques of Bezier path.

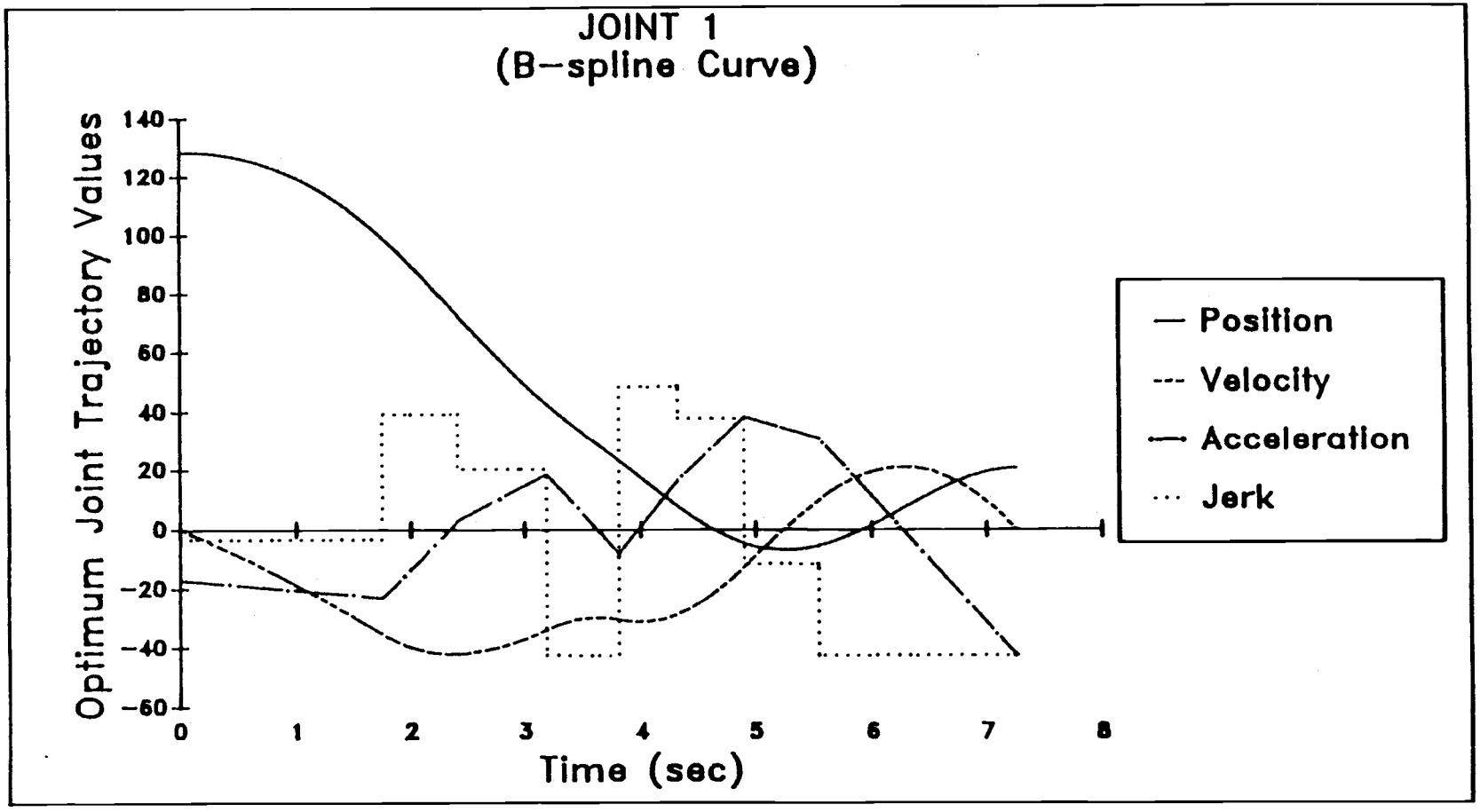


Figure 6.6: Optimum joint trajectories of B-spline path for joint 1.

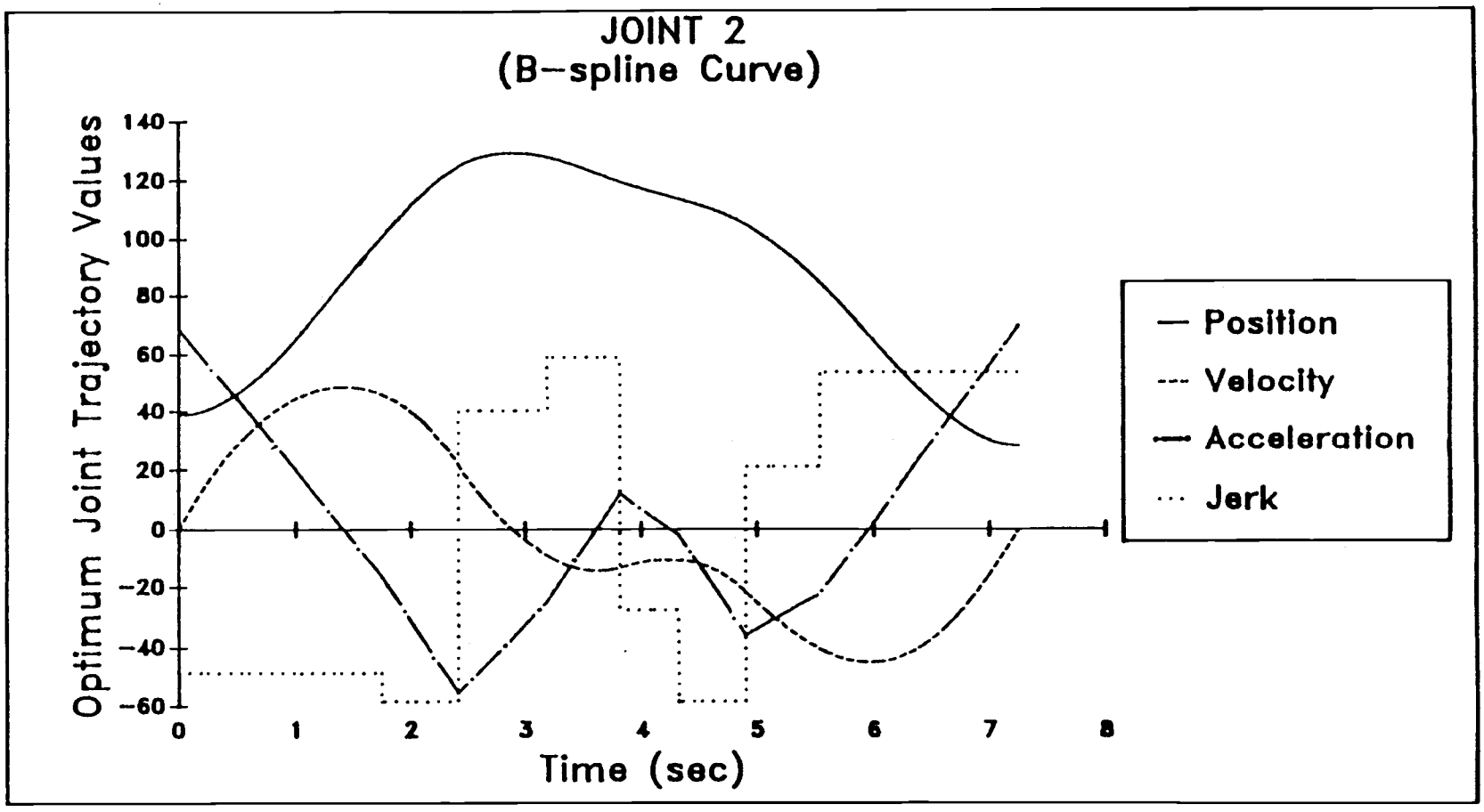


Figure 6.7: Optimum joint trajectories of B-spline path for joint 2.

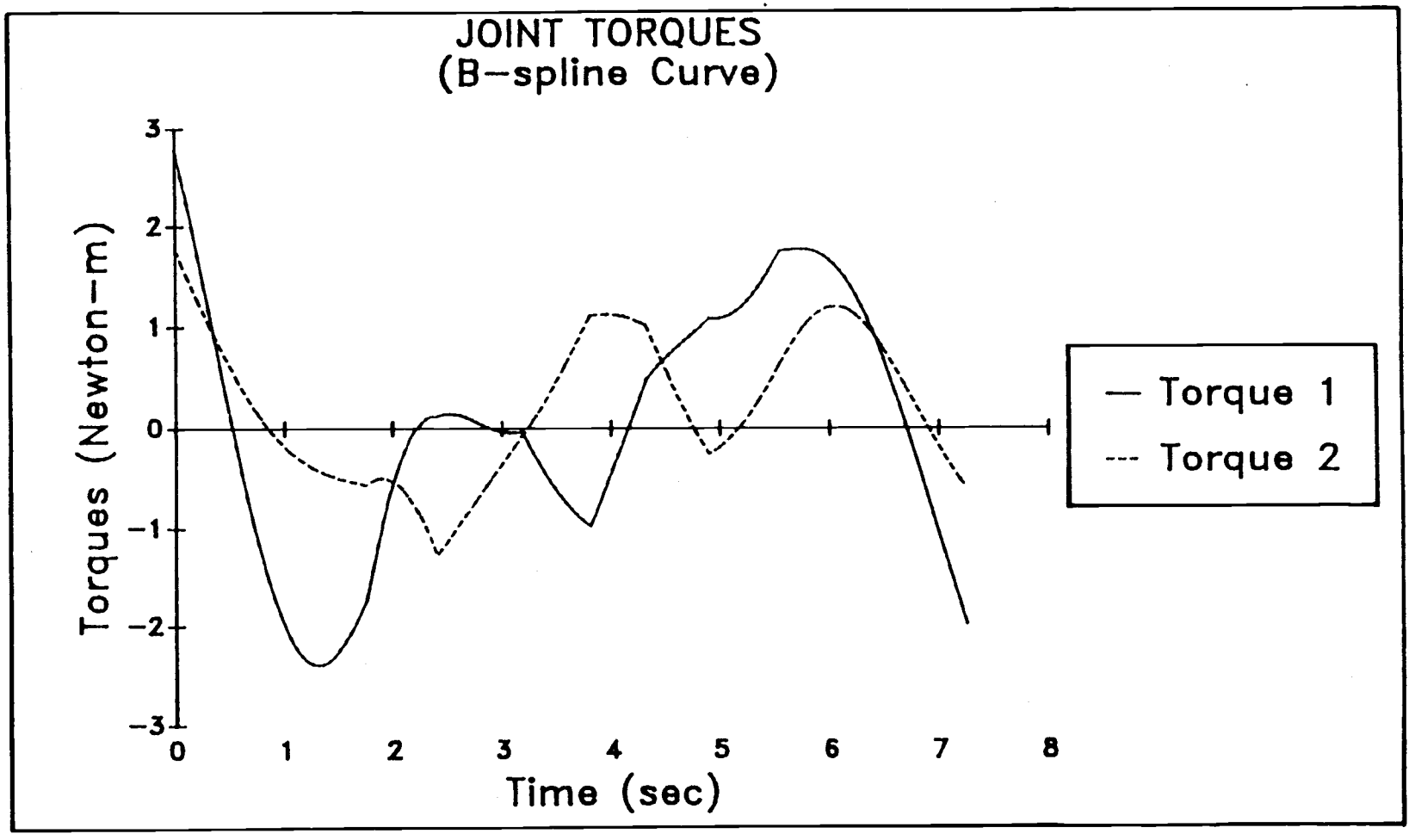


Figure 6.8: Optimum joint torques of B-spline path.

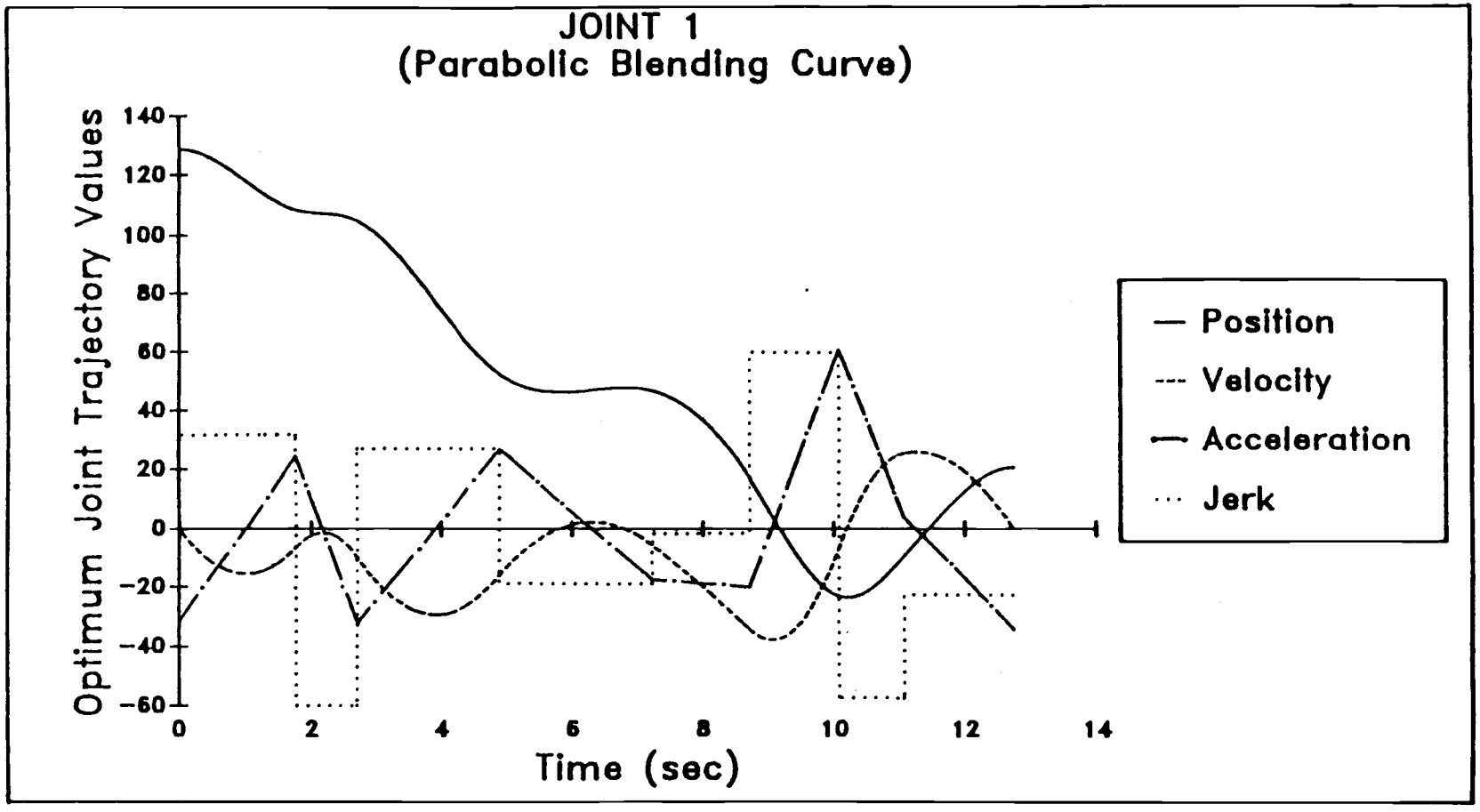


Figure 6.9: Optimum joint trajectories of parabolic blending path for joint 1.

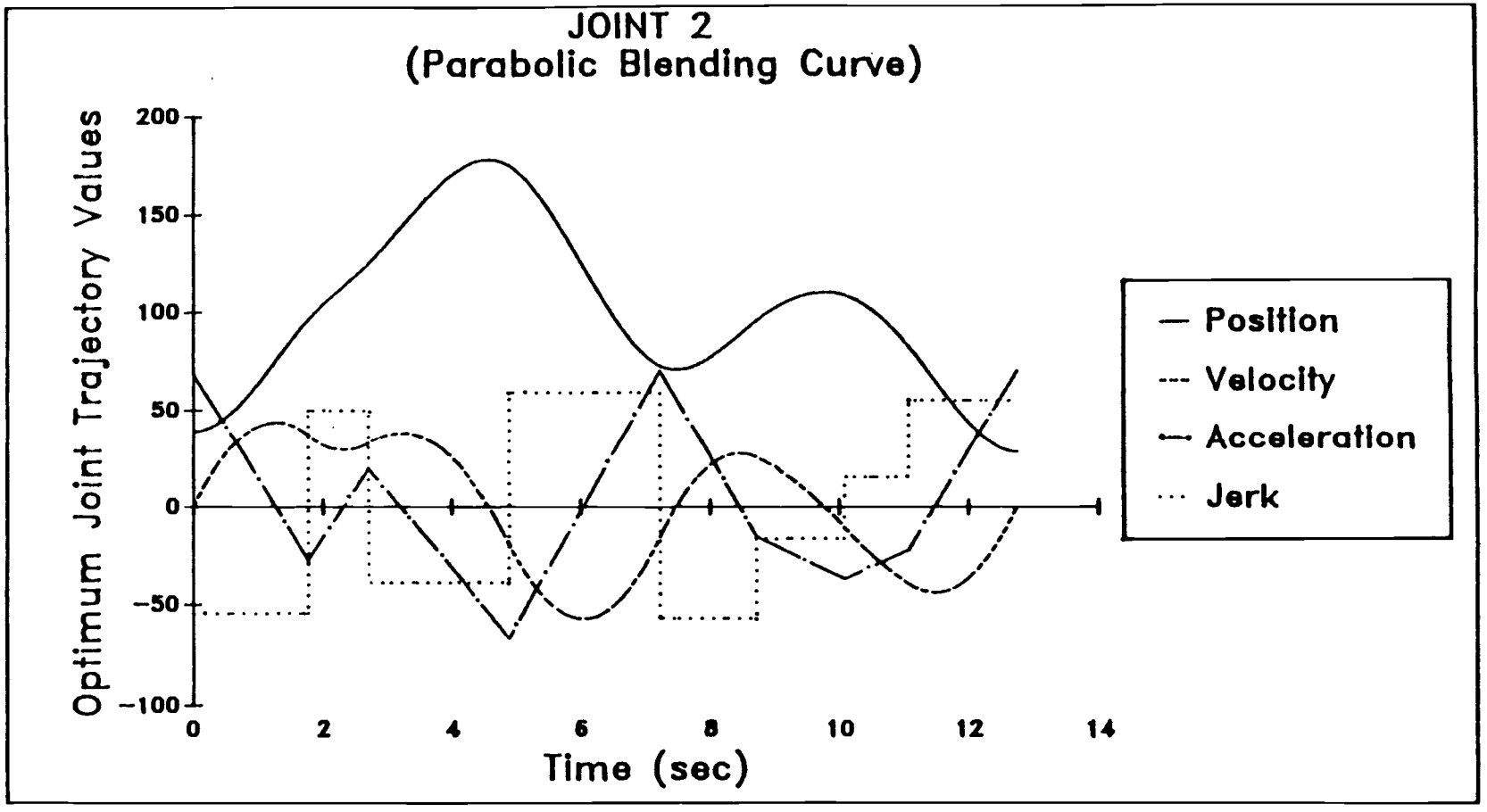


Figure 6.10: Optimum joint trajectories of parabolic blending path for joint 2.

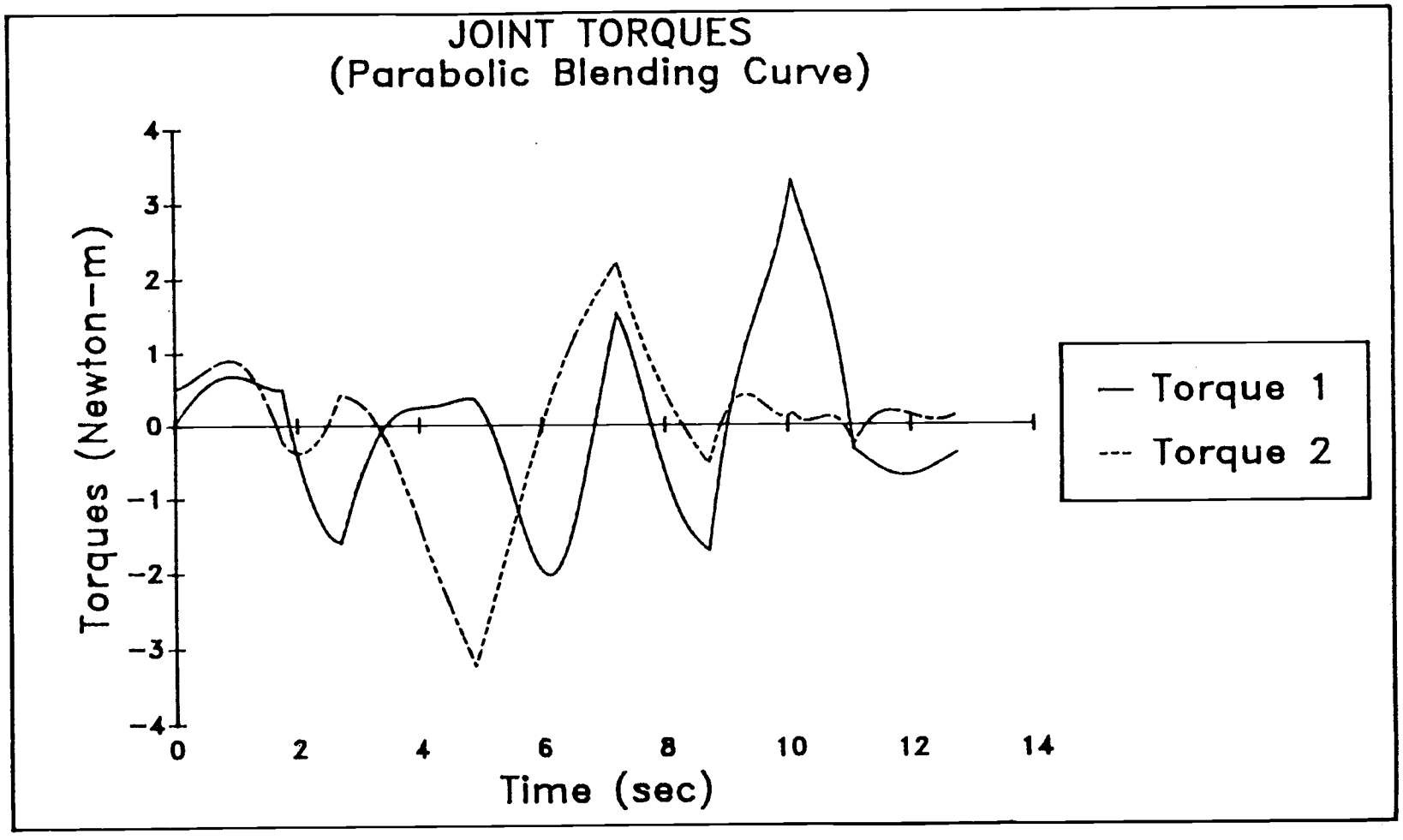


Figure 6.11: Optimum joint torques of parabolic blending path.

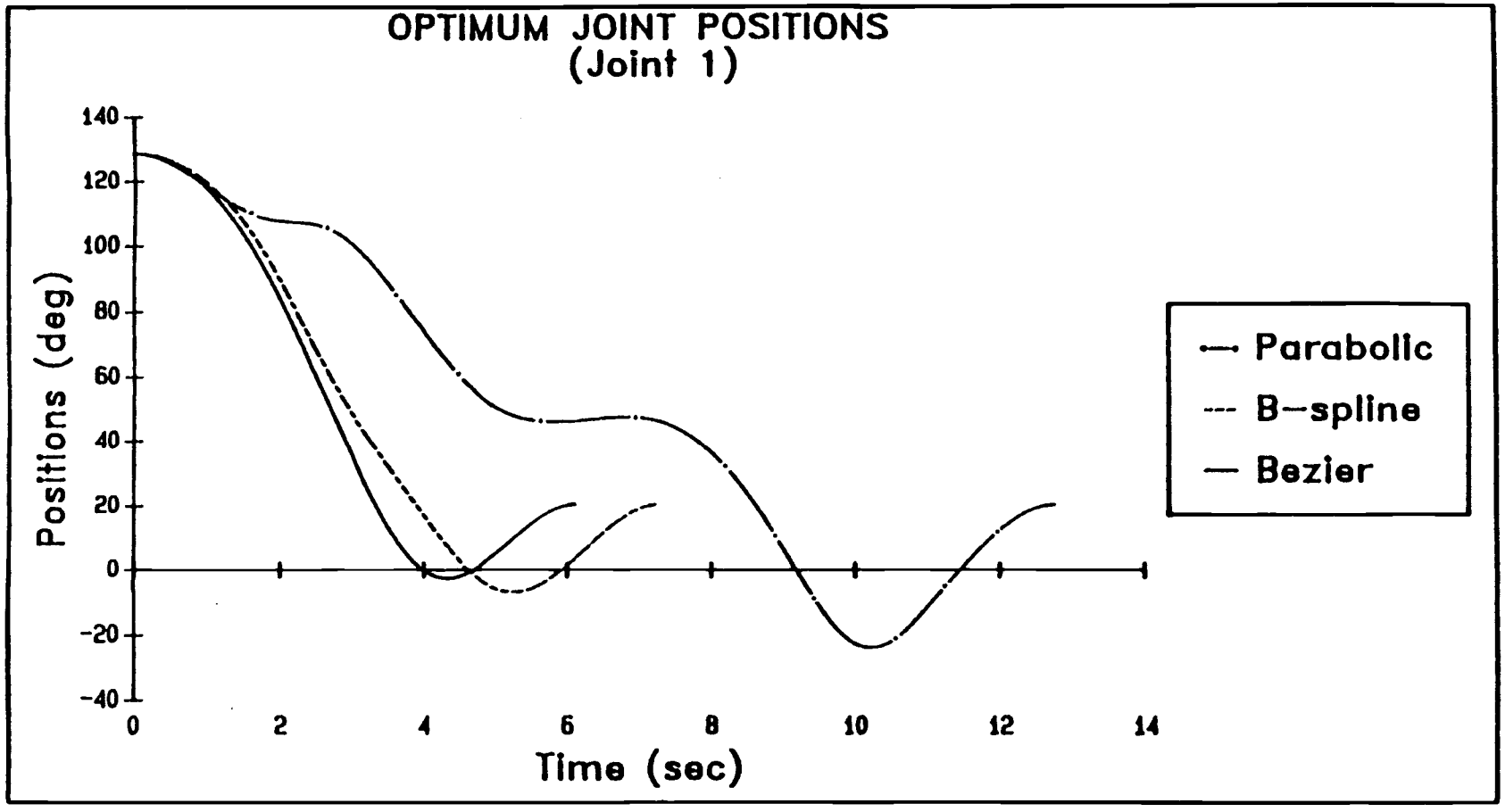


Figure 6.12: Optimum joint positions for joint 1.

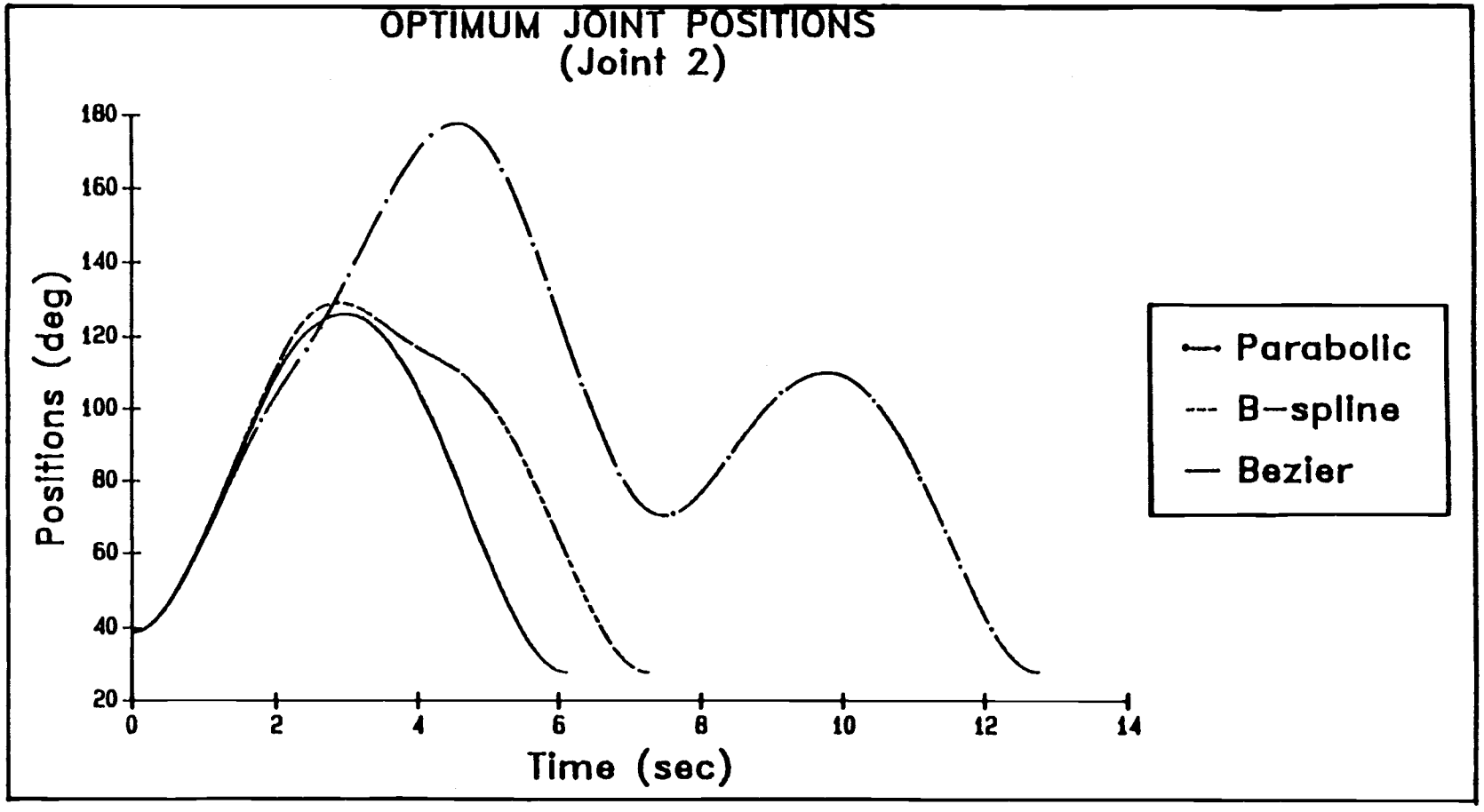


Figure 6.13: Optimum joint positions for joint 2.

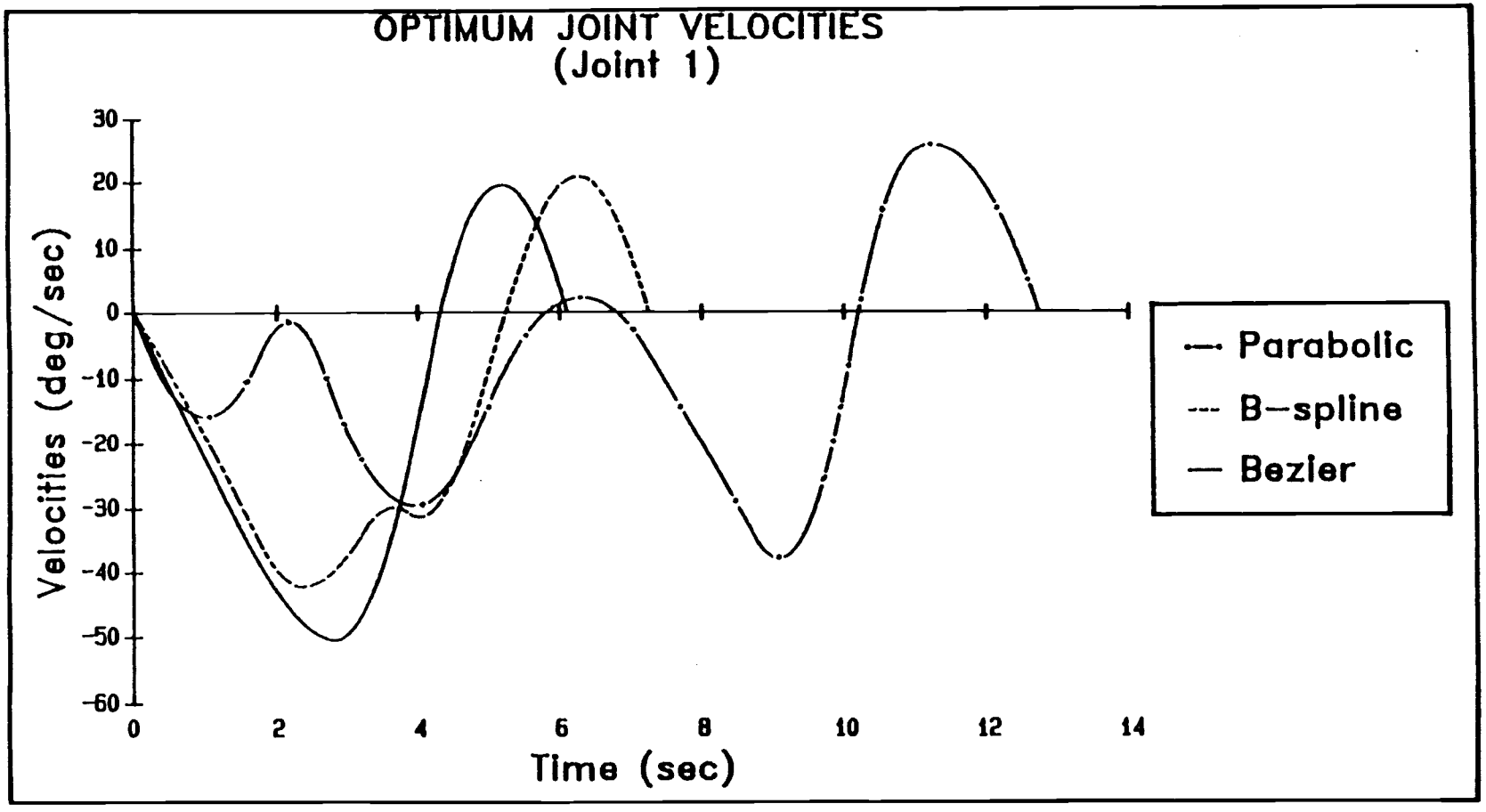


Figure 6.14: Optimum joint velocities for joint 1.

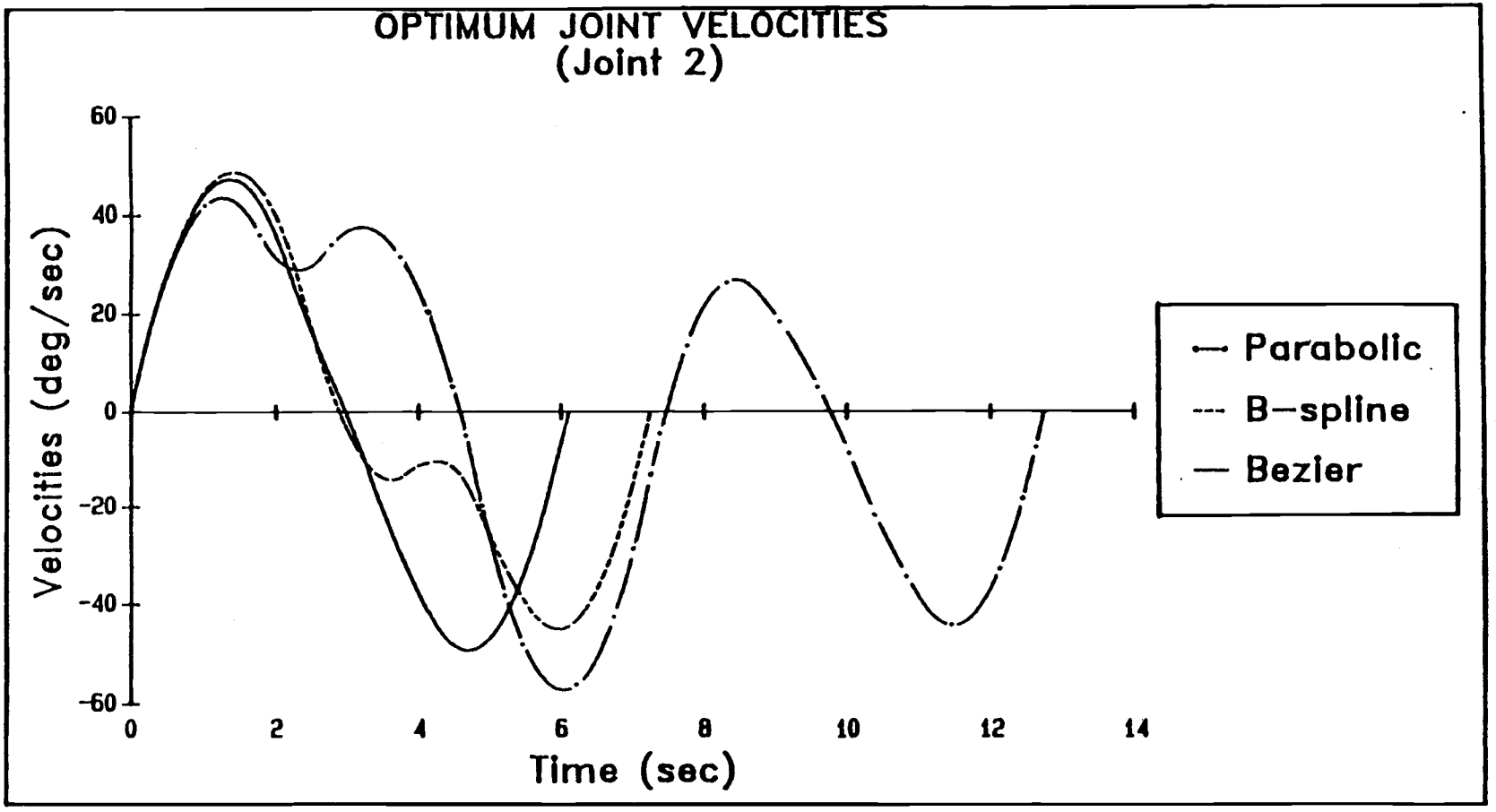


Figure 6.15: Optimum joint velocities for joint 2.

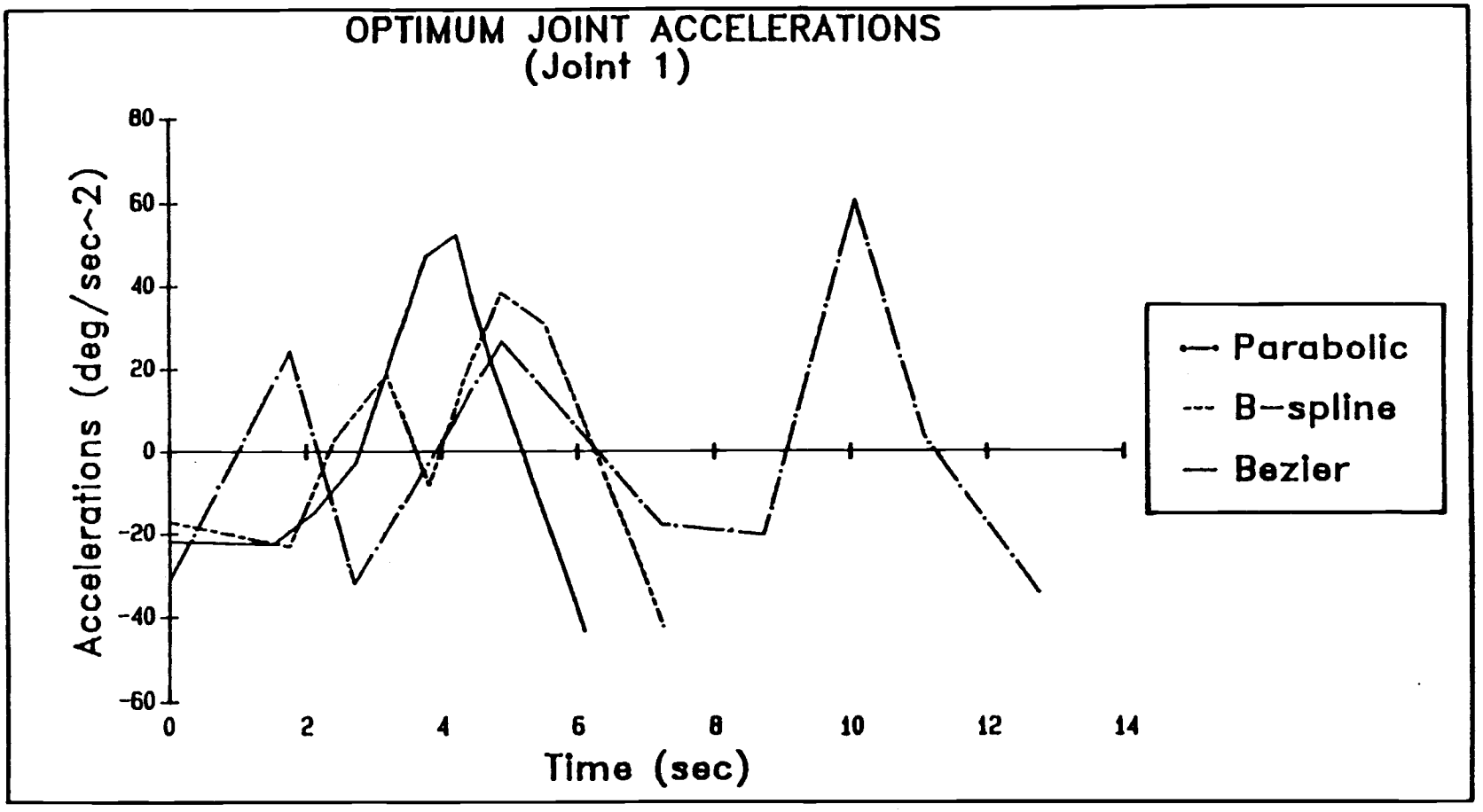


Figure 6.16: Optimum joint accelerations for joint 1.

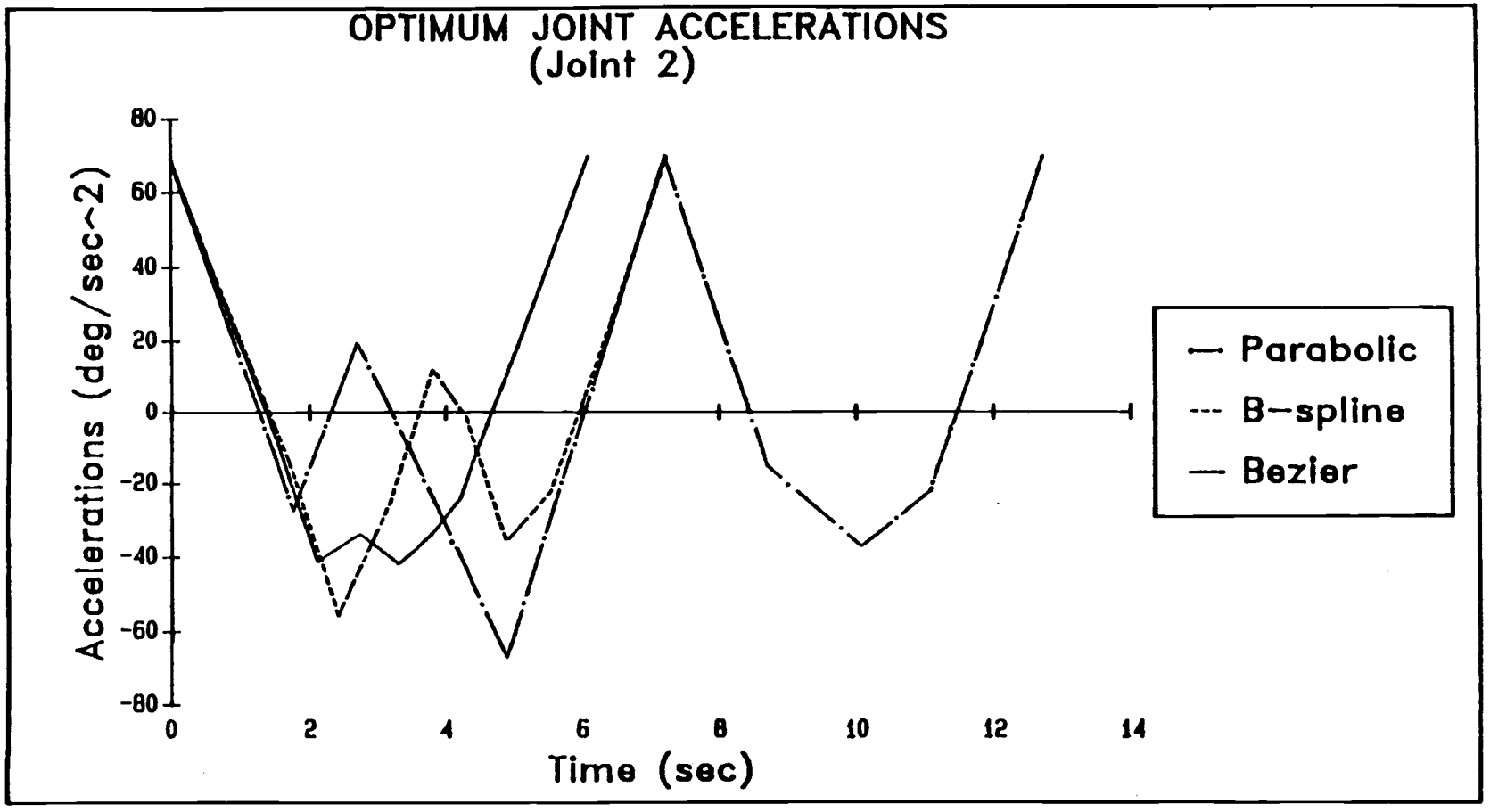


Figure 6.17: Optimum joint accelerations for joint 2.

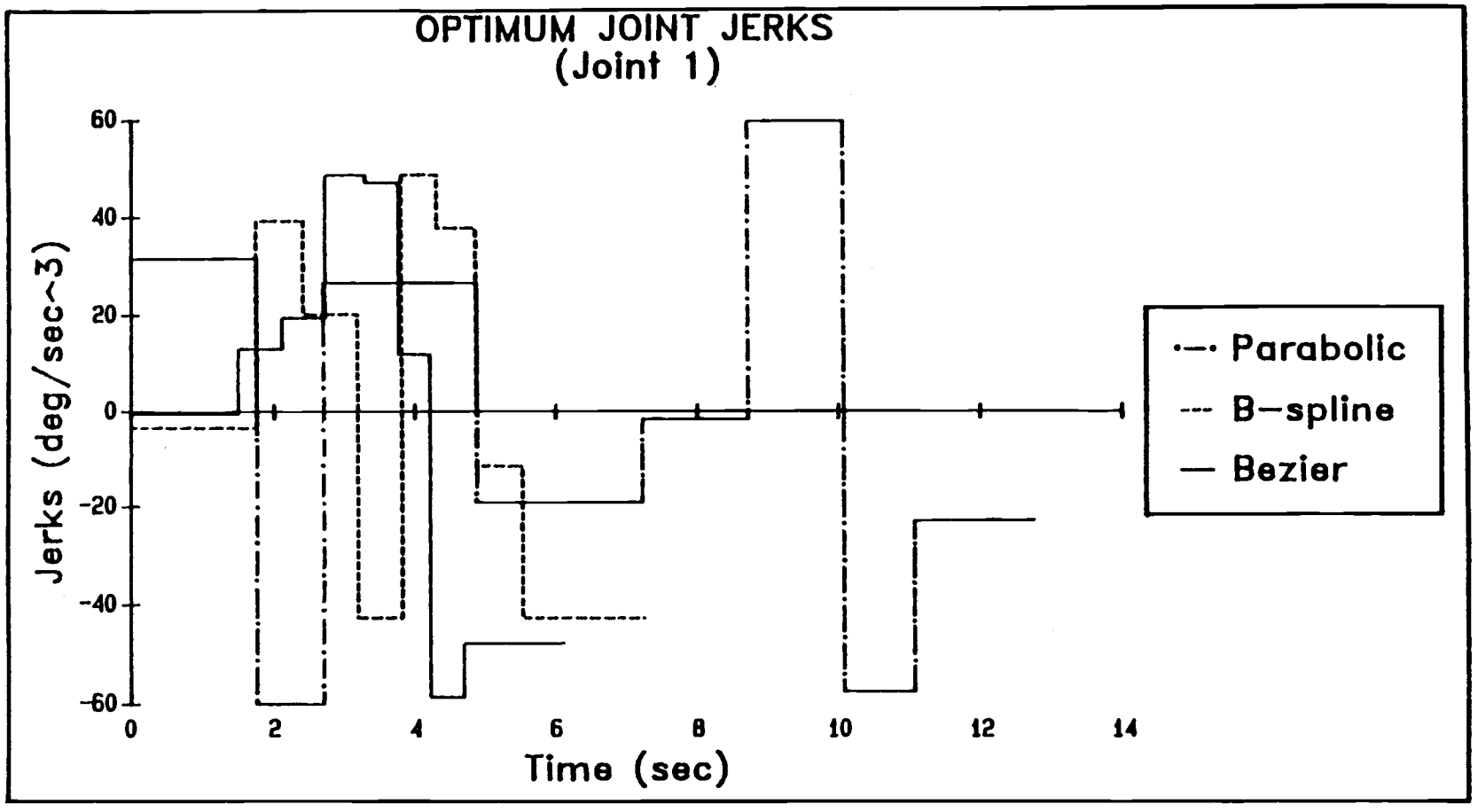


Figure 6.18: Optimum joint jerks for joint 1.

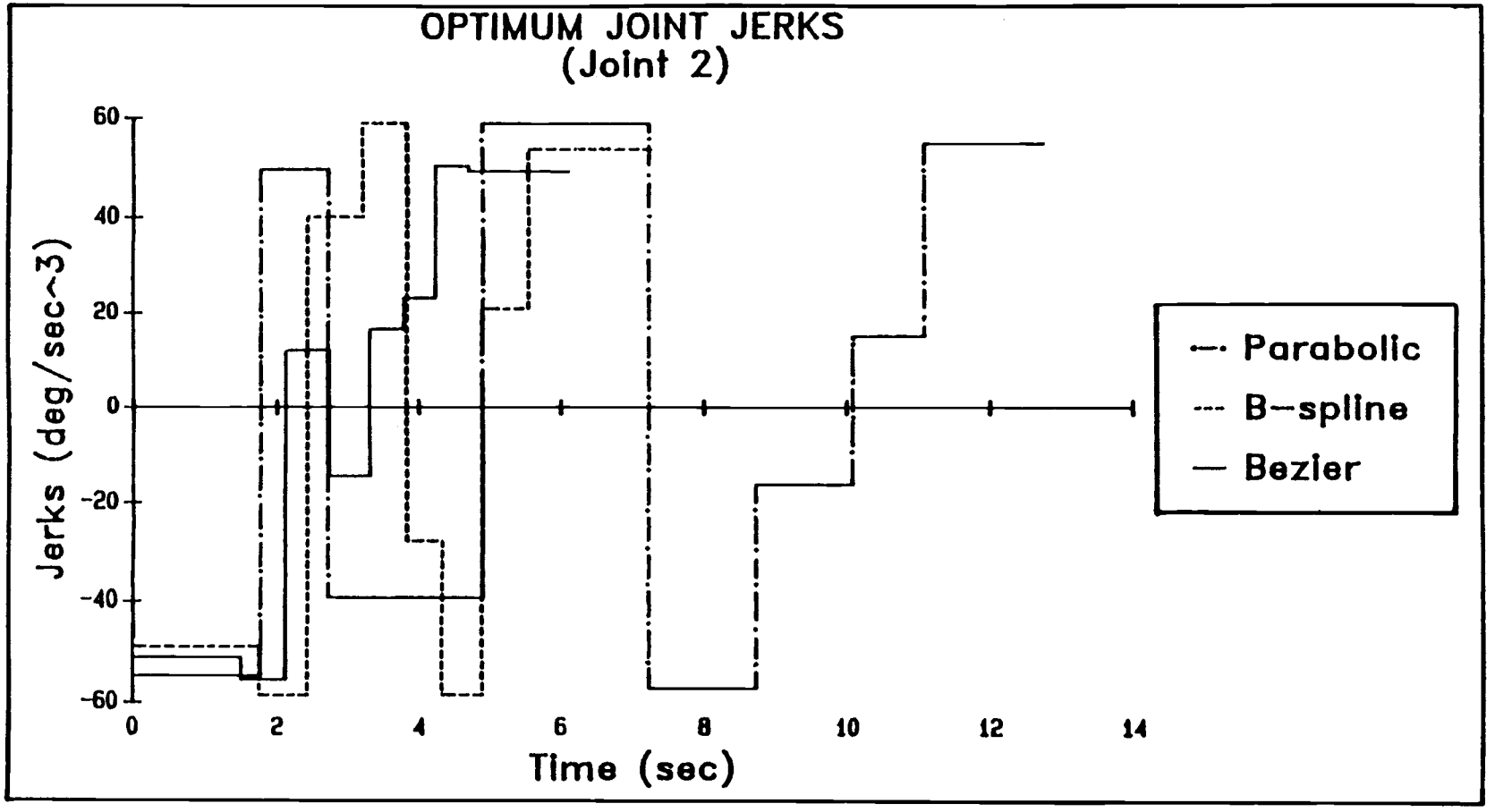


Figure 6.19: Optimum joint jerks for joint 2.

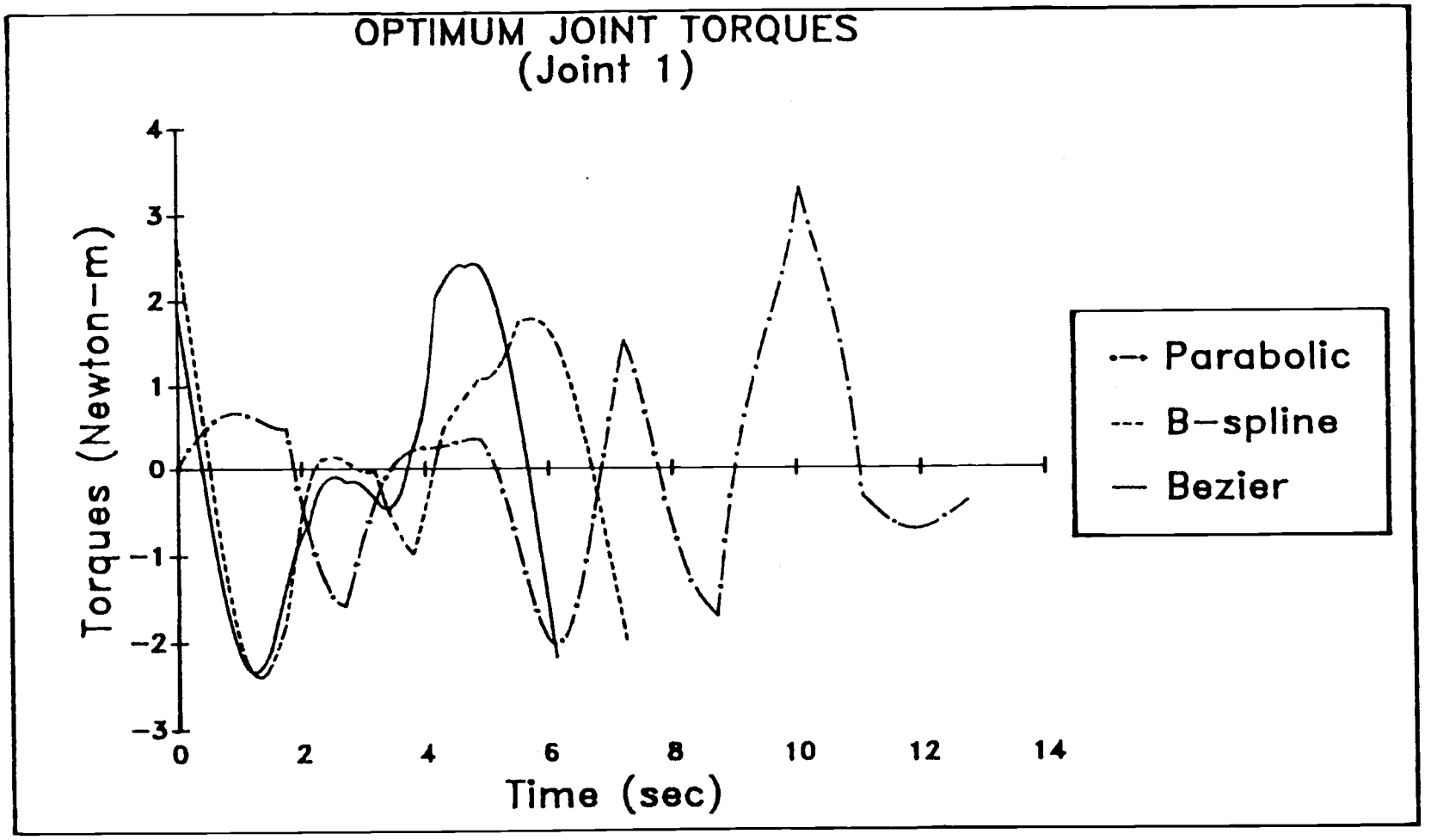


Figure 6.20: Optimum joint torques for joint 1.

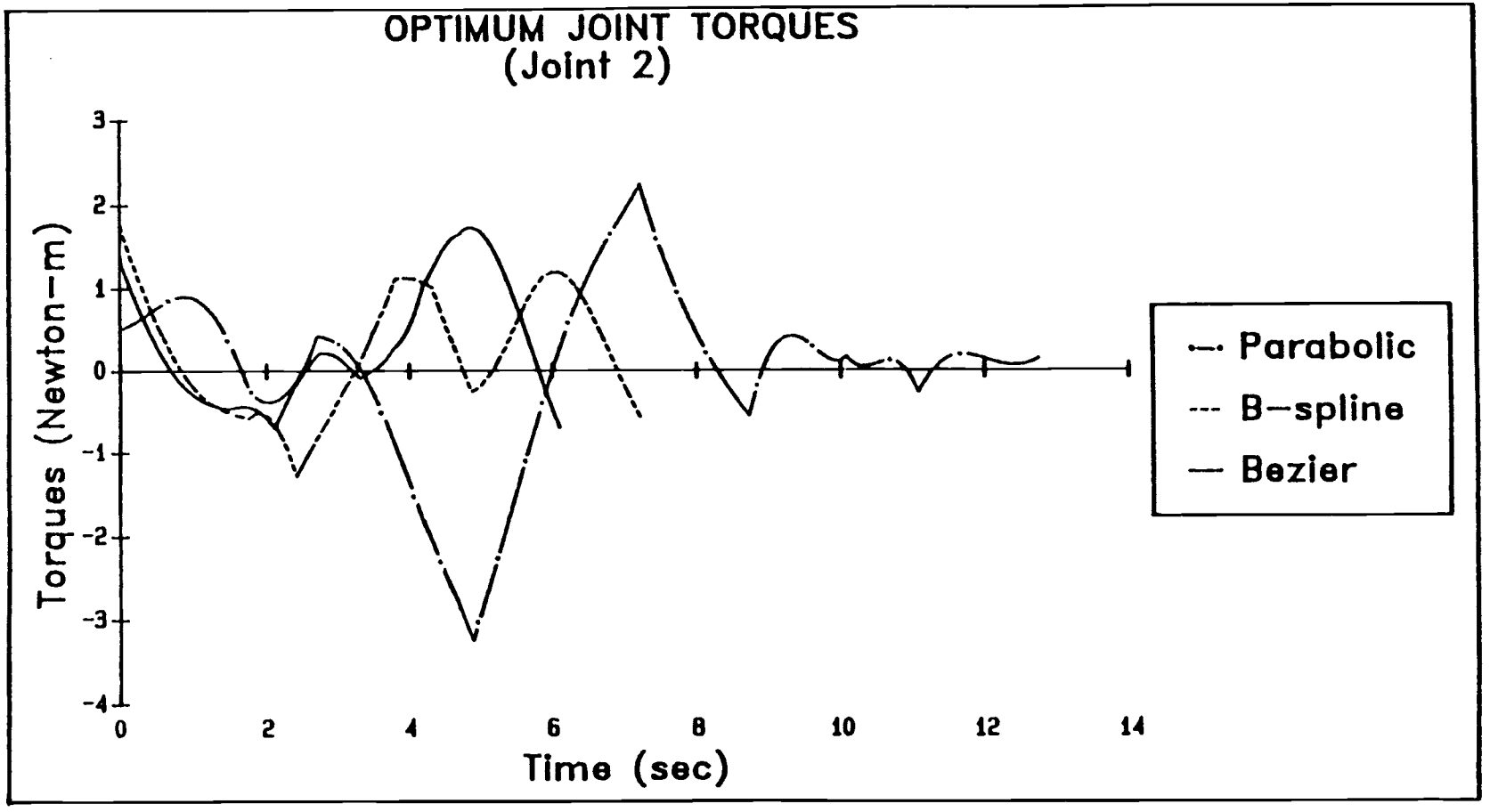


Figure 6.21: Optimum joint torques for joint 2.

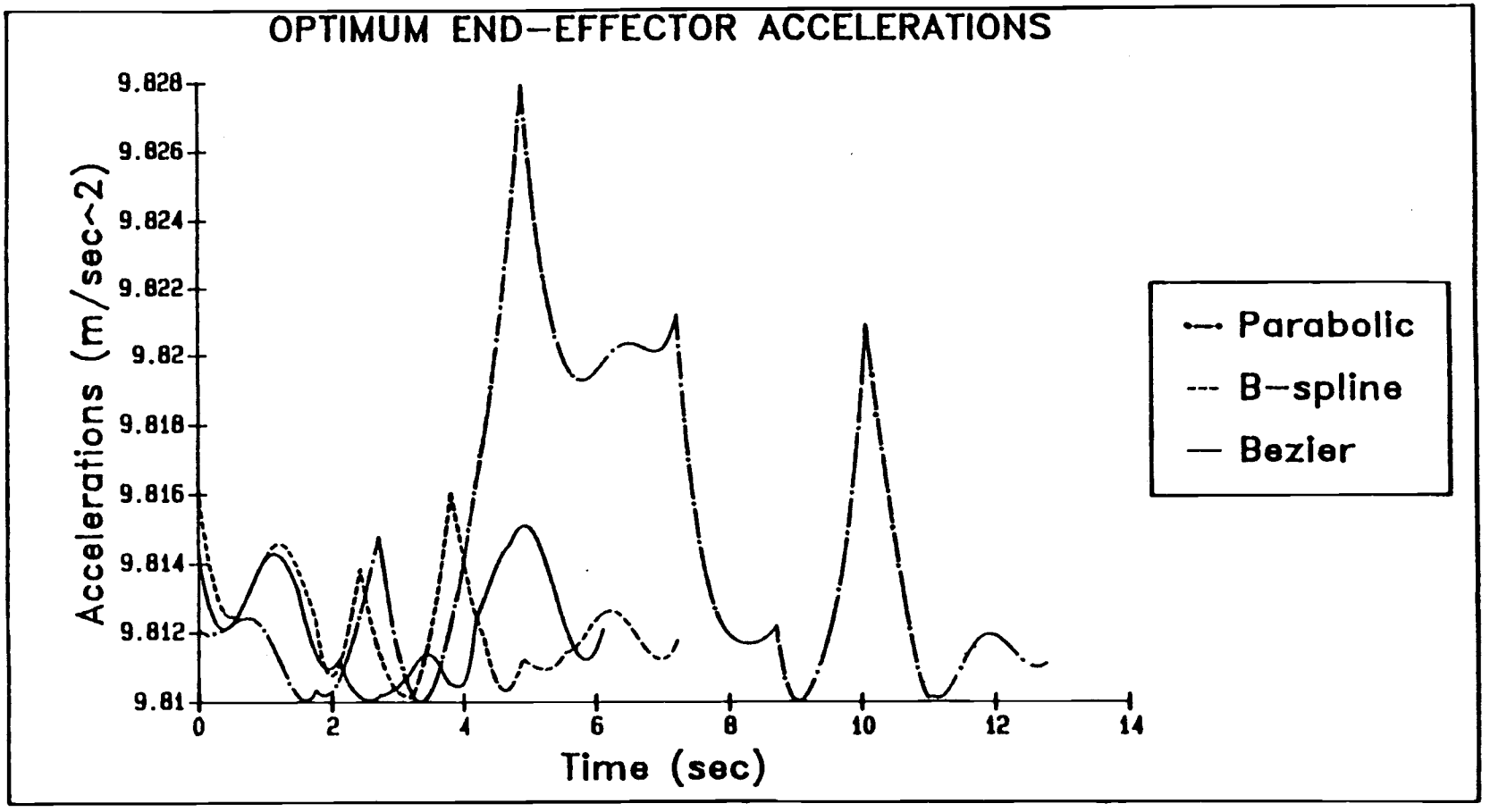


Figure 6.22: Optimum end-effector accelerations.

TABLE 7. Comparison of end-effector paths in normalized time.

End-effector Paths \ Curve Methods	Bezier Curve	B-spline Curve (k=4)	Parabolic Blending Curve
1	4.846 = 1	4.734 = 0.977	5.695 = 1.175
2	5.610 = 1	5.519 = 0.984	8.378 = 1.493
3	6.662 = 1	8.141 = 1.222	11.747 = 1.763
4	6.105 = 1	7.252 = 1.188	12.743 = 2.087
5	6.019 = 1	6.091 = 1.012	9.408 = 1.563
6	5.997 = 1	7.589 = 1.265	10.231 = 1.706

The result of this comparison shows that the approximation curve techniques (Bezier and B-spline (k=4) curves) gave shorter traveling time than the interpolation curve technique (parabolic blending curve).

6.2.2 EXAMPLE 2

In the second type of example, we will apply three curve techniques to construct Cartesian end-effector paths of the two degrees of freedom manipulator for different sets of initial time estimates. The purpose of this example is to find how consistently the initially guessed time estimates affect the total traveling time of a robot task.

We applied the same three curve methods for two different shapes of end-effector paths in six different sets of initial time intervals $h_1, h_2, \ldots, h_{n-1}$. Figure 6.23 shows one of the two end-effector paths in two dimensional view. The other end-effector path is shown in Appendix C. The end-effector path is predefined by five control points in Cartesian space. Referring to Figure 6.23, the manipulator tip starts

at rest at point $P_0(-0.95,0.2)$ and stops at point $P_4(0.9,0.4)$ by moving through the other points $P_1(-0.5,0.2)$, $P_2(-0.1,0.8)$, and $P_3(0.4,0.8)$, respectively. Applying the technique as in Example 1, each curve technique generates Cartesian knot points along the end-effector path as shown in Table 8 and corresponding joint displacements with one set for each joint are shown in Table 9.

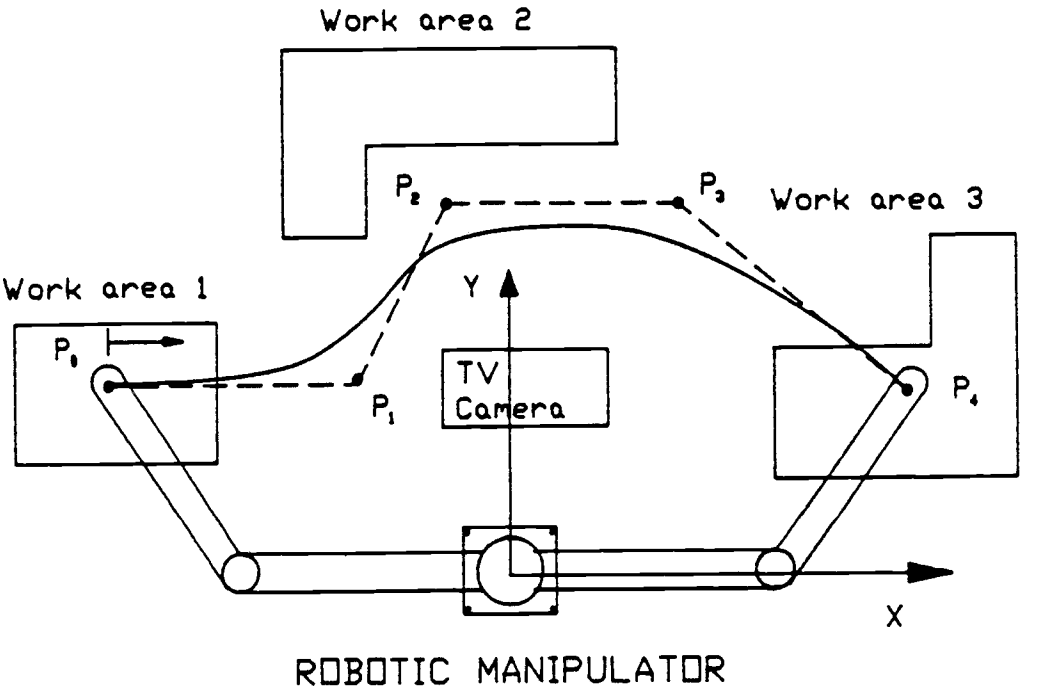


Figure 6.23: The top view of a 2 DOF manipulator on its intuitive path (Example 2).

TABLE 8. Cartesian knots along end-effector path (in meters).

Curves / Knots \ Coor.	Bezier		B-spline (k=4)		Parabolic blending	
	X	Y	X	Y	X	Y
1	-0.950	0.200	-0.950	0.200	-0.950	0.200
2	-0.729	0.247	-0.655	0.249	-0.705	0.171
3	-0.510	0.355	-0.428	0.369	-0.500	0.200
4	-0.290	0.481	-0.243	0.516	-0.232	0.403
5	-0.066	0.588	-0.075	0.650	-0.100	0.800
6	0.166	0.648	0.102	0.730	0.126	0.836
7	0.405	0.643	0.309	0.731	0.400	0.800
8	0.651	0.561	0.568	0.629	0.673	0.657
9	0.900	0.400	0.900	0.400	0.900	0.400

TABLE 9. Joint displacements for Cartesian knots (in degrees).

Curves / Knots \ Joint	Bezier		B-spline (k=4)		Parabolic blending	
	Joint 1	Joint 2	Joint 1	Joint 2	Joint 1	Joint 2
1	154.24	27.75	154.24	27.75	154.24	27.75
2	121.56	79.40	113.67	91.00	122.83	87.06
3	93.60	103.08	83.67	111.19	100.78	114.83
4	65.32	111.65	60.04	110.38	57.57	124.61
5	42.61	107.52	47.45	98.26	60.85	72.54
6	27.61	96.03	39.58	84.95	49.16	64.55
7	17.24	81.07	29.63	74.88	36.87	53.13
8	10.02	61.51	15.85	64.08	24.41	38.80
9	13.99	19.95	13.99	19.95	13.99	19.95

The optimization results of Bezier, B-spline and parabolic blending curve methods are presented for different sets of initially guessed time intervals in Tables 10, 11 and 12, respectively.

TABLE 10. The optimization results of Bezier curve for total traveling time.

Time Intervals (sec)	1		2		3		4		5		6	
	I	F	I	F	I	F	I	F	I	F	I	F
h_1	3	1.560	2.5	1.576	6.4	1.557	2.7	1.601	6.5	1.567	3.6	1.554
h_2	3	0.623	2.9	0.621	5.8	0.614	3.3	0.626	5.4	0.628	3.2	0.612
h_3	3	0.623	3.5	0.616	5.2	0.570	4.8	0.596	4.1	0.620	4.7	0.552
h_4	3	0.561	3.9	0.606	4.7	0.525	5.9	0.575	2.9	0.619	4.2	0.507
h_5	3	0.507	4.4	0.726	4.2	0.433	5.1	0.576	3.5	0.488	5.6	0.413
h_6	3	0.486	4.8	0.566	3.6	0.377	4.3	0.459	4.6	0.420	3.9	0.388
h_7	3	0.486	5.3	0.503	2.9	0.422	3.7	0.455	5.3	0.454	7.8	0.430
h_8	3	1.418	6.2	1.452	2.6	1.332	3.1	1.351	6.1	1.383	5.3	1.335
Total Time	24	6.263	33.5	6.667	35.4	5.829	32.9	6.238	38.4	6.180	38.3	5.790

I: Initial

F: Final

TABLE 11. The optimization results of B-spline curve (k=4) for total traveling time.

Time Intervals (sec)	1		2		3		4		5		6	
	I	F	I	F	I	F	I	F	I	F	I	F
h_1	3	1.828	2.5	1.826	6.4	1.843	2.7	1.811	6.5	1.813	3.6	1.816
h_2	3	0.707	2.9	0.663	5.8	0.785	3.3	0.667	5.4	0.667	3.2	0.674
h_3	3	0.755	3.5	0.616	5.2	0.748	4.8	0.655	4.1	0.650	4.7	0.606
h_4	3	0.655	3.9	0.588	4.7	1.099	5.9	0.677	2.9	0.576	4.2	0.586
h_5	3	0.767	4.4	0.582	4.2	0.621	5.1	0.670	3.5	0.538	5.6	0.541
h_6	3	0.538	4.8	0.658	3.6	0.592	4.3	0.650	4.6	0.608	3.9	0.651
h_7	3	0.633	5.3	0.729	2.9	0.705	3.7	0.717	5.3	0.710	7.8	0.738
h_8	3	2.292	6.2	1.709	2.6	1.814	3.1	1.759	6.1	1.759	5.3	1.703
Total Time	24	8.176	33.5	7.371	35.4	8.207	32.9	7.607	38.4	7.321	38.3	7.315

I: Initial

F: Final

TABLE 12. The optimization results of parabolic blending curve for total traveling time.

Time Intervals (sec)	1		2		3		4		5		6	
	I	F	I	F	I	F	I	F	I	F	I	F
h_1	3	1.679	2.5	1.659	6.4	1.666	2.7	1.691	6.5	1.691	3.6	1.669
h_2	3	0.611	2.9	0.629	5.8	0.626	3.3	0.627	5.4	0.583	3.2	0.634
h_3	3	1.423	3.5	1.221	5.2	1.205	4.8	1.193	4.1	2.270	4.7	1.190
h_4	3	1.500	3.9	1.737	4.7	1.629	5.9	1.635	2.9	1.250	4.2	1.840
h_5	3	1.437	4.4	0.901	4.2	0.925	5.1	1.133	3.5	3.203	5.6	0.950
h_6	3	1.017	4.8	0.892	3.6	1.021	4.3	0.801	4.6	0.967	3.9	0.900
h_7	3	0.621	5.3	0.617	2.9	0.635	3.7	0.592	5.3	0.602	7.8	1.143
h_8	3	1.094	6.2	1.087	2.6	1.097	3.1	1.079	6.1	1.083	5.3	1.220
Total Time	24	9.381	33.5	8.744	35.4	8.803	32.9	8.749	38.4	11.649	38.3	9.546

I: Initial

F: Final

According to the above tables, initially guessed time intervals $(h_1,h_2,\ldots,h_8)$ consistently affect the total traveling time of the manipulator. However, the effect makes only one second difference in overall total traveling time. In other words, it shows the local and global optimum results.

CHAPTER 7

SUMMARY, CONCLUSIONS, DISCUSSIONS AND FUTURE WORK

7.1 SUMMARY

This thesis presents a method for constructing minimum-time robot trajectories between given end states in a workspace containing obstacles. The method is preferably applied to the geometric collision-free paths of a manipulator. Bezier, B-spline (k=4), and parabolic blending curves are used for end-effector path planning by using a sequence of control points in Cartesian space. Thus, our objective has been to concentrate on the minimum-time path planning problem of both Cartesian and joint paths and their numerical solution. The approach adopted has consisted in controlling the collision-free robot path at joint level after conversion of the end-effector knot points along the path into joint displacements.

The algorithm is developed to construct end-effector paths using polynomial curve techniques in Cartesian space. Then each curve technique generates knot points along the end-effector path for the given Cartesian control points. Therefore, the motion of the manipulator is specified by a sequence of position and orientation knot points of the end-effector. These knots are transformed into sets of joint displacements, with one set for each joint. Cubic splines are used to fit the sequence of joint displacements for each joint if time intervals between each pair of adjacent knots are given. An optimization algorithm is developed in Chapter 4 to minimize efficiently the total traveling time of the manipulator by adjusting all the time intervals among knots with constraints on joint positions, velocities,

accelerations, jerks, torques, and end-effector acceleration. Also, this method considers the full non-linear dynamics of the manipulator.

7.2 CONCLUSIONS AND DISCUSSIONS

The numerical computation for the examples in Chapter 6 was easily accomplished. However, it may not prevent the robot arm from colliding with the obstacle at points in its workspace other than the tip of the robot arm. For this case, each joint trajectory generated may be transformed to the Cartesian space to check that it avoids the obstacles in the workspace. Therefore, the algorithm in this study should be improved into a more sophisticated algorithm to obtain the collision-free robot trajectories.

In approximation curve methods, Bezier and B-spline curves, as more vertices are used for the end-effector path representation, the extra computation time will be required for the optimization of joint trajectories. The total traveling time is greatly reduced with only a few vertices, and the addition of more vertices only slightly decreases the total traveling time.

The advantage of B-spline curves is their local approximation basis which is particularly useful for collision avoidance schemes. For the B-splines, it appears that the optimum end-effector path may converge as the number of path degrees of freedom are increased.

The high local path curvature may require the manipulator to slow down in order to stay on the path. Therefore, this may result in local minima for the optimization. In the optimization of joint trajectories, the question "Have the robot trajectories converged to the global

minimum?" is very difficult to answer since this would require us to search through a space of many parameters.

Due to the complexity of the problem, the algorithm requires much computer computation time. Computational efficiency was not the main concern of this study. However, the algorithm can be used for general off-line programming robot applications when the robot equations of motion are available and the geometric description of the robot workspace is known. From our experiments, the path planning program, ROBOPATH, can also provide information about the performance of a manipulator, help to locate the critical physical limits of a given robot design, and help to define the manipulator's tasks and work cells.

7.3 FUTURE WORK

The algorithm in this thesis should be improved into a more sophisticated algorithm to obtain the collision-free robot trajectories and no-jerk trajectories. It should be extended to the case of six degrees of freedom manipulators. In addition, it should be extended to have a Computer-Aided Design (CAD) implementation for manipulator motion in the workspace of the manipulator. Graphics computation capability can also be added into ROBOPATH to plot the time optimal joint trajectories for output. B-spline functions can be used for constructing the joint trajectories to get continuous jerk function. This algorithm might be modified for the robot arm with flexible linkages.

BIBLIOGRAPHY

Ahlberg, J.H., Nilson, E.N., and Walsh, J.L. 1967. The Theory of Splines and Their Applications. Academic Press, New York.

Asada, H. and Slotine, J.-J.E. 1986. Robot Analysis and Control. John Wiley and Sons, New York.

Barsky, B.A. 1984 (Jan.). A Description and Evaluation of Various 3-D Models. IEEE Computer Graphics, pp. 38-52.

Bobrow, J.E. 1982. Optimal Control of Manipulators. Ph.D. Thesis, University of California Los Angeles.

Bobrow, J.E., Dubowsky, S., and Gibson, J.S. 1983 (June). On the Optimal Control of Robotic Manipulators with Actuators Constraints. Proc. of the 1983 ACC, San Francisco, California, pp. 782-787.

Bobrow, J.E., Dubowsky, S., and Gibson, J.S. 1985 (Fall). Time-optimal Control of Robotic Manipulators Along Specified Paths. Int. J. Robotics Res., vol. 4, no. 3, pp. 3-17.

Brady, M.T., Hollerbach, M., Johnson, T.L., Lozano-Perez, T., and Mason, M.T. 1982. Robot Motion: Planning and Control. MIT Press, Cambridge, Massachusetts.

Braibant, V. and Geradin, M. 1987 (Oct.-Dec.). Optimum Path Planning of Robot Arms. Robotica, vol. 5, part 4, pp. 323-331.

Chand, S. and Doty, K.L. 1985 (Summer). On-line Polynomial Trajectories for Robot Manipulators. Int. J. Robotics Res., vol. 4, no. 2, pp. 38-48.

Cheney, W. and Kincaid, D. 1985. Numerical Mathematics and Computing. Brooks/Cole Publishing Company, Monterey, Califiornia, pp. 231-234.

Cox, M.G. 1972. The Numerical Evaluation of B-splines. J. Inst. Maths. Applics., vol. 10, pp. 134-149.

Craig, C. 1986. Introduction to Robotics: Mechanics and Control. Addison-Wesley Publishing, Reading, Massachusetts.

de Boor, C. 1972. On Calculating with B-splines. Journal of Approximation Theory, vol. 6, pp. 50-62.

de Boor, C. 1978. A Practical Guide to Splines. Springer-Verlag, App. Math. Sci., vol. 27, New York.

Dubowsky, S. and Shiller, Z. 1984 (June). Optimal Dynamic Trajectories for Robotic Manipulators. Proc. of the V CISM-IFToMM Symposium on the Theory and Practice of Robots and Manipulators, Udine, Italy.

Dubowsky, S., Norris, M.A., and Shiller, Z. 1986. Time Optimal Trajectory Planning for Robotic Manipulators with Obstacle Avoidance: a CAD Approach. IEEE Int. Conf. on Robotics and Automation, pp. 1906-1912.

Edwall, C.W., Ho, C.Y., and Pottinger, H.J. 1982 (March). Trajectory Generation and Control of a Robot Arm Using Spline Functions. SME Technical Paper, MS82-218, pp. 1-23.

Faux, I.D. and Pratt, M.J. 1985. Computational Geometry for Design and Manufacture. Ellis Horwood, New York.

Fu, K.S., Gonzales, R.C., and Lee, C.S.G. 1987. Robotics: Control, Sensing, Vision, and Intelligence. McGraw-Hill, New York.

Goldenberg, A.A. and Lawrence, D.L. 1986 (June). End Effector Path Generation. Trans. ASME, J. Dyn. Syst. Meas. Contr., vol. 108, pp. 158-162.

Himmelblau, D.M. 1972. Applied Nonlinear Programming. McGraw-Hill, New York, pp. 142-148.

Hooke, R. and Jeeves, T.A. 1961. Direct Search Solution of Numerical and Statistical Problems. J. of the Assn. for Computing Machinery, vol. 8, pp. 212-229.

Jacoby, S.L.S., Kowalik, J.S., and Pizzo, J.T. 1972. Iterative Methods for Nonlinear Optimization Problems. Prentice-Hall, Inc., Englewood Cliffs, New Jersey, pp. 71-73.

Jeon, H.T. and Eslami, M. 1986. A Minimum Time Joint Trajectory Planning for Industrial Manipulator with Input Torque Constraint. IEEE Int. Conf. on Robotics and Automation, pp. 559-564.

Johnson, D.W. and Gilbert, E.G. 1985 (Dec.). Minimum Time Robot Path Planning in the Presence of Obstacles. IEEE Proc. of 24th Conf. on Decision and Control, pp. 1748-1753.

Kim, B.K. and Shin, K.G. 1984. An Efficient Minimum-Time Robot Path Planning Under Realistic Conditions. IEEE Proc. ACC, pp. 296-303.

Kim, B.K. and Shin, K.G. 1985 (Mar./Apr.). Minimum-Time Path Planning for Robot Arms and Their Dynamics. IEEE Trans. Sys. Man Cyber., vol. SMC-15, no. 2, pp. 213-223.

Lin, C.S. and Chang, P.R. 1982 (Nov.). Joint Trajectory Generation of Mechanical Manipulators Using Spline Functions. SME Technical Paper, MS82-503, pp. 1-10.

Lin, C.S. and Chang, P.R. 1985. Approximate Optimum Paths of Robot Manipulators Under Realistic Physical Constraints. IEEE Int. Conf. on Robotics and Automation, pp. 737-742.

Lin, C.S., Chang, P.R., and Luh, J.Y.S. 1983 (Dec.). Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots. IEEE Trans. Auto. Contr., vol. AC-28, no. 12, pp. 1066-1074.

Luh, J.Y.S. and Lin, C.S. 1981 (June). Optimum Path Planning for Mechanical Manipulators. ASME Trans. J. Dynam. Syst. Meas. Contr., vol. 102, pp. 142-151.

Luh, J.Y.S. and Lin, C.S. 1984 (May/June). Approximate Joint Trajectories for Control of Industrial Robots Along Cartesian Paths. IEEE Trans. Sys. Man Cyber., vol. SMC-14, no. 3, pp. 444-450.

Luh, J.Y.S. and Walker, M.W. 1977. Minimum-Time Along the Path for a Mechanical Arm. IEEE Proc. of 16th Conf. on Decision and Control, pp. 755-759.

Luh, J.Y.S., Walker, M.W., and Paul, R.P. 1980 (June). On-Line Computational Scheme for Mechanical Manipulators. Trans. ASME, J. Dyn. Syst. Meas. Contr., vol. 102, pp. 69-76.

Maus, R. and Allsup, R. 1986. Robotics: A Manager's Guide. John Wiley and Sons, New York.

Mizugaki, Y., Kimura, F., Sata, T., and Suzuki, T. 1985. Generation of a Free Curve Trajectory with a Specified Velocity Distribution for an Articulated Robot. Design and Synthesis, pp. 569-574.

Mortenson, M.E. 1985. Geometric Modeling. John Wiley and Sons, New York.

Newman, W.W. and Sproull, R.F. 1979. Principles of Interactive Computer Graphics. 2nd ed., McGraw-Hill, New York.

Park, H.Y. 1988. Nonlinear Multi-Objectives Optimization Using the Modified Goal Programming. M.S. Project, Department of Mechanical Engineering, Oregon State University.

Paul, R.P. 1979 (Nov.). Manipulator Cartesian Path Control. IEEE Trans. Sys. Man Cyber., vol. SMC-9, no. 11, pp. 702-711.

Paul, R.P. and Zhang, H. 1985. Robot Motion Trajectory Specification and Generation. Robotics Research 2, pp. 187-194, MIT Press, Cambridge, Massachusetts.

Pike, R.W. 1986. Optimization for Engineering Systems. Van Nostrand Reinhold, New York.

Powell, M.J.D. 1981. Approximation Theory and Methods. Cambridge University Press, London.

Rajan, V.T. 1985. Minimum Time Trajectory Planning. IEEE Int. Conf. on Robotics and Automation, pp. 759-764.

Rogers, D.F. and Adams, J.A. 1976. Mathematical Elements for Computer Graphics. McGraw-Hill, New York.

Sahar, G. and Hollerbach, J.M. 1985. Planning of Minimum-Time Trajectories for Robot Arms. IEEE Int. Conf. on Robotics and Automation, pp. 751-758.

Schumaker, L.L. 1981. Spline Functions: Basic Theory. John Wiley and Sons, New York.

Seeger, G.H. and Paul, R.P. 1985. Optimizing Robot Motion Along a Predefined Path. IEEE Int. Conf. Robotics and Automation, pp. 765-770.

Shiller, Z. 1984 (June). Optimal Dynamic Trajectories and Modeling of Robotic Manipulators. M.S. Thesis, Massachusetts Institute of Technology.

Shiller, Z. and Dubowsky, S. 1985 (March). On the Optimal Control of Robotic Manipulators with Actuator and End-Effector Constraints. IEEE Intern. Conf. on Robotics and Automation, St. Louis, Missouri, pp. 614-620.

Shin, K.G. and McKay, N.D. 1983 (Dec.). An Efficient Robot Arm Control Under Geometric Path Constraints. IEEE Proc. of 22nd Conf. on Decision and Control, vol. 3, pp. 1449-1457.

Shin, K.G. and McKay, N.D. 1984. Robot Path Planning Using Dynamic Programming. IEEE Proc. of 23rd Conf. on Decision and Control, vol. 3, pp. 1629-1635.

Shin, K.G. and McKay, N.D. 1985 (June). Minimum-Time Control of Robotic Manipulators with Geometric Path Constraints. IEEE Trans. Automatic Control, vol. AC-30, no. 6, pp. 531-541.

Shin, K.G. and McKay, N.D. 1986a (June). A Dynamic Programming Approach to Trajectory Planning of Robotic Manipulators. IEEE Trans. Automatic Control, vol. AC-31, no. 6, pp. 491-500.

Shin, K.G. and McKay, N.D. 1986b (June). Selection of Near-Minimum Time Geometric Paths for Robotic Manipulators. IEEE Trans. Automatic Control, vol. AC-31, no. 6, pp. 501-511.

Shin, K.G. and McKay, N.D. 1986c. Minimum-Time Trajectory Planning for Industrial Robots with General Torque Constraints. IEEE Int. Conf. on Robotics and Automation, vol. 1, pp. 412-417.

Thompson, S.E. and Patel, R.V. 1985 (Sept.). Formulation of Joint Trajectories for Industrial Robots Using B-splines. IEEE COMPINT 85: Computer Aided Technologies: Proceedings, pp. 484-488.

Vukobratovic, M. and Kircanski, M. 1982 (June). A Method for Optimal Synthesis of Manipulation Robot Trajectories. Trans. ASME, J. Dyn. Syst. Meas. Contr., vol. 104, pp. 188-193.

APPENDICES

APPENDIX A

CONVENTION FOR LINK COORDINATE SYSTEMS AND PARAMETERS

In order to describe the location of each link relative to its neighbors, we use Craig's convention and define a frame attached to each link. A frame (coordinate system) must be set up in each link according to the rules given below.

A.1 LINK PARAMETERS

Since the link frames have been attached to the links according to convention in Craig (1986), the following definitions of the link parameters are valid:

a_i (link length) : the distance from Z_i to Z_{i+1} measured along X_i ;

α_i (link twist) : the angle between Z_i and Z_{i+1} measured about X_i ;

d_i (link offset) : the distance from X_{i-1} to X_i measured along Z_i ;

θ_i (joint variable) : the angle between X_{i-1} and X_i measured about Z_i .

a_i is usually chosen greater than zero since it corresponds to a distance; however, α_i, d_i, and θ_i are signed quantities.

A.2 FIRST AND LAST LINKS

The frame {0}, attached to the base of the manipulator, cannot move and can be considered the reference frame. Since frame {0} is arbitrary, it always simplifies matters to choose Z_0 along axis 1. To locate frame {0} it coincides with frame {1} when joint variable θ_1 is

zero. It is always true that $a_0 = 0.0$ and $\alpha_0 = 0.0$ by using this convention.

If joint N is revolute, the direction of X_N is chosen to be aligned with X_{N-1} when $\theta_N = 0.0$, and the origin of frame {N} is chosen so that $d_N = 0.0$. If joint N is prismatic, the direction of X_N is chosen to have $\theta_N = 0.0$, and the origin of frame {N} is chosen at the intersection of X_{N-1} and joint axis N when $d_N = 0.0$.

A.3 INTERMEDIATE LINKS

The Z-axis of frame {i}, Z_i, is coincident with the joint axis i. The origin of frame {i} is located where the a_i perpendicular intersects the joint i axis. X_i points along a_i in the direction from joint i to joint i+1. Y_i is formed by the right hand rule to complete the ith frame.

APPENDIX B

MANIPULATOR JACOBIANS

B.1 VELOCITY PROPAGATION FROM LINK TO LINK

The velocity of link i+1 is that of link i plus a new rotational component was added by joint i+1. The angular velocity of link i+1 is the same as that of link i plus a new component caused by rotational velocity at joint i+1.

We can express the description of the angular velocity of link i+1 with respect to frame {i+1} in the following form:

$$ {}^{i+1}\omega_{i+1} = {}^{i+1}_{i}R\,{}^{i}\omega_{i} + \dot{\theta}_{i+1}\,{}^{i+1}\hat{Z}_{i+1} \quad . \tag{B.1} $$

We can compute the linear velocity of link i+1 with respect to frame {i+1}:

$$ {}^{i+1}\vartheta_{i+1} = {}^{i+1}_{i}R\,({}^{i}\vartheta_{i} + {}^{i}\omega_{i} \times {}^{i}P_{i+1}) \quad . \tag{B.2} $$

Since the velocities are desired in terms of the base coordinate system, they can be rotated into base coordinates by multiplication with ${}^{0}_{N}R$.

We would like to calculate the velocity of the tip of the manipulator as a function of joint rates with respect to the nonmoving base frame. Since velocity vector is a frame vector, we may rotate ${}^{3}\vartheta_{3}$ and ${}^{3}\omega_{3}$ with the rotation matrix ${}^{0}_{3}R$, which is

$$ {}^{0}_{3}R = {}^{0}_{1}R\,{}^{1}_{2}R\,{}^{2}_{3}R = \begin{bmatrix} c_{12} & -s_{12} & 0 \\ s_{12} & c_{12} & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad . \tag{B.3} $$

Sequentially from link to link, we calculate:

$$ {}^1\omega_1 = {}^1_0R\ {}^0\omega_0 + \dot{\theta}_1\ {}^1\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1 \end{bmatrix} \quad ; \qquad (B.4) $$

$$ {}^1\vartheta_1 = {}^1_0R\ ({}^0\vartheta_0 + {}^0\omega_0 x\ {}^0P_1) = \begin{bmatrix} 0 \\ 0 \\ 0 \end{bmatrix} \quad ; \qquad (B.5) $$

$$ {}^2\omega_2 = {}^2_1R\ {}^1\omega_1 + \dot{\theta}_1\ {}^1\hat{Z}_1 = \begin{bmatrix} 0 \\ 0 \\ \dot{\theta}_1+\dot{\theta}_2 \end{bmatrix} \quad ; \qquad (B.6) $$

$$ {}^2\vartheta_2 = {}^2_1R\ ({}^1\vartheta_1 + {}^1\omega_1 x\ {}^1P_2) = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 \\ 0 \end{bmatrix} \quad ; \qquad (B.7) $$

Since θ_3 is always zero,

$$ {}^3\omega_3 = {}^2\omega_2 \quad ; \qquad (B.8) $$

Since $ {}^3_2R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} $,

$$ {}^3\vartheta_3 = {}^3_2R\ ({}^2\vartheta_2 + {}^2\omega_2 \times {}^2P_3) = \begin{bmatrix} l_1 s_2 \dot{\theta}_1 \\ l_1 c_2 \dot{\theta}_1 + l_2(\dot{\theta}_1+\dot{\theta}_2) \\ 0 \end{bmatrix} . \qquad (B.9) $$

Since we want to compute the velocity of the tip of the robot arm with respect to the robot base frame, we rotate it with the rotation matrix ${}^0_3R$. This rotation yields

$$ {}^0\dot{\theta}_3 = {}^0_3R\ {}^3\dot{\theta}_3 = \begin{bmatrix} -l_1 s_1 \dot{\theta}_1 - l_2 s_{12}(\dot{\theta}_1+\dot{\theta}_2) \\ l_1 c_1 \dot{\theta}_1 + l_2 c_{12}(\dot{\theta}_1+\dot{\theta}_2) \\ 0 \end{bmatrix} \quad . \qquad \text{(B.10)} $$

B.2 JACOBIANS

The manipulator Jacobians which are time-varying linear transformations represent the relationship between the joint displacements and the end-effector location at the present position and robot arm configuration. The elements of the Jacobian are functions of joint displacements, and therefore vary with the robot arm configuration. In the field of robotics, the velocity relationship between the joints and the end-effector (the tip) of the manipulator is determined by the manipulator Jacobians (Craig 1986; Asada and Slotine 1986). We can write it as:

$$ {}^0\nu = {}^0J(\underline{\theta})\dot{\underline{\theta}} \quad , \qquad \text{(B.11)} $$

where $\underline{\theta}$ is the vector of joint angles of the manipulator, and ν is a vector of Cartesian velocities. In Equ. (B.11), the leading superscript for Jacobian notation indicates in which frame the resulting Cartesian velocity is expressed. For the general case of an N jointed robot, the Jacobian is NxN, $\dot{\underline{\theta}}$ is Nx1, and ${}^0\nu$ is Nx1. The number of rows

in a Jacobian equals the number of degrees of freedom in the Cartesian space being considered. The number of columns in a Jacobian is equal to the number of joints of the manipulator.

In our case of the two-link manipulator, we may write a 2x2 Jacobian which relates joint rates to end-effector velocity. The Jacobian may be written in frame {0} is seen to be:

$$ {}^{0}J(\underline{\theta}) = \begin{bmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{bmatrix} . \qquad (B.12) $$

This square matrix relates joint rates to end-effector velocity. We could also consider the 3x2 Jacobian which would include the angular velocity of the end-effector.

B.3 SINGULARITIES

If the Jacobian matrix is nonsingular, then it can be inverted to calculate joint rates for given Cartesian velocities:

$$ \dot{\underline{\theta}} = J^{-1}(\underline{\theta})\, \nu \qquad (B.13) $$

We must determine whether the Jacobian is invertible for all values of $\underline{\theta}$ or not. If not, where is it not invertible? Most manipulators have values of $\underline{\theta}$ where the Jacobian becomes singular; such locations are called **singularities**. Each manipulator has singularities at the boundary of its workspace, and most manipulators have loci of singularities inside their workspace. When the manipulator is in a singular configuration, it has lost one or more degrees of freedom as viewed from Cartesian space. In other words, there is some subspace in Cartesian space along which it is impossible to move the end-effector of

the robot no matter which joint rates are selected. This happens at the workspace boundary of robots.

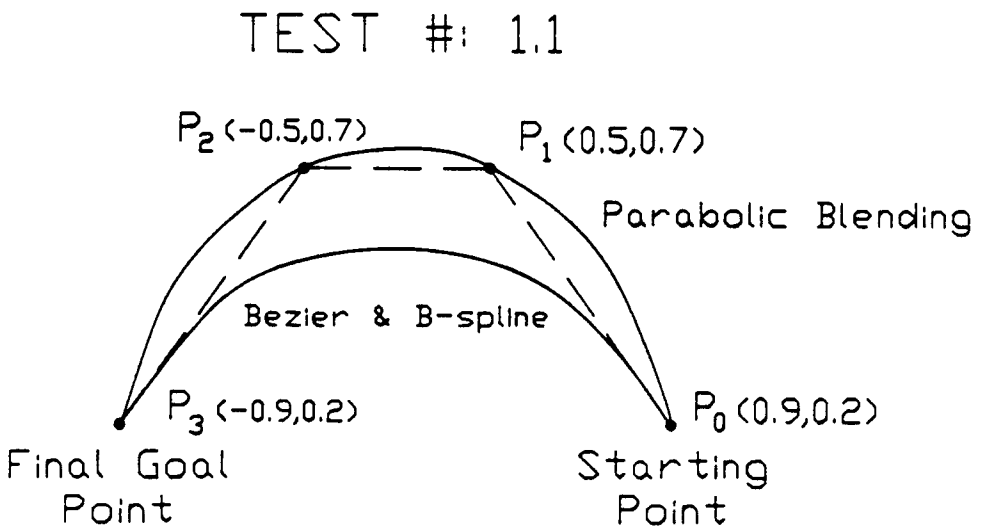
To find the singular points of our two-link manipulator, we must examine the determinant of its Jacobian. Where the determinant is equal to zero, the Jacobian is singular.

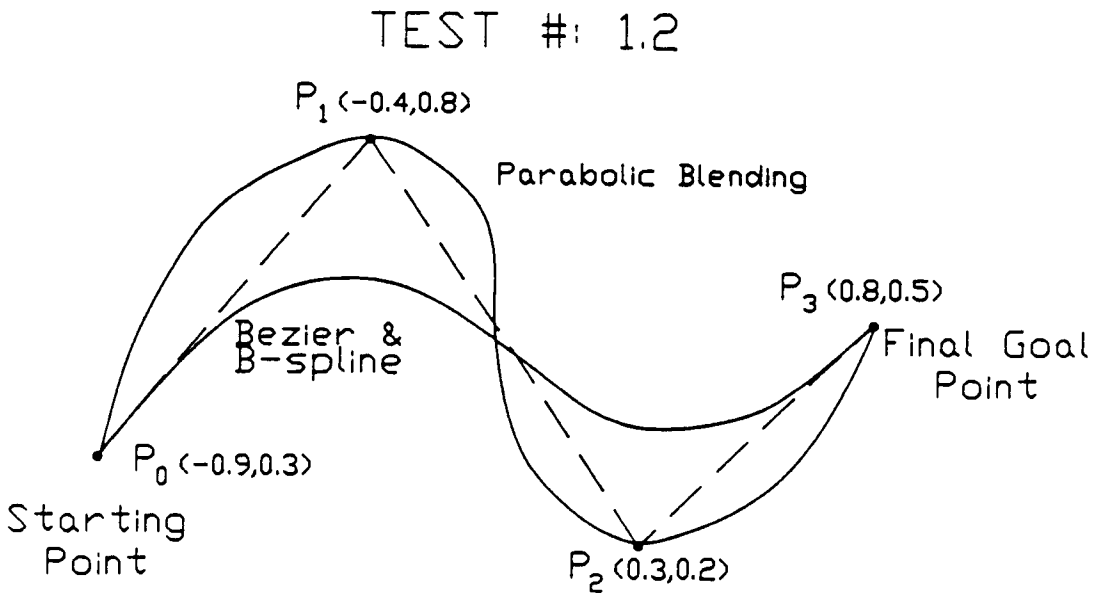
$$\text{DET}[J(\underline{\theta})] = \begin{vmatrix} -l_1 s_1 - l_2 s_{12} & -l_2 s_{12} \\ l_1 c_1 + l_2 c_{12} & l_2 c_{12} \end{vmatrix} = l_1 l_2 s_2 = 0 \ . \quad \text{(B.14)}$$

It is obvious that a singularity of the manipulator exists when θ_2 is 0 or 180 degrees.

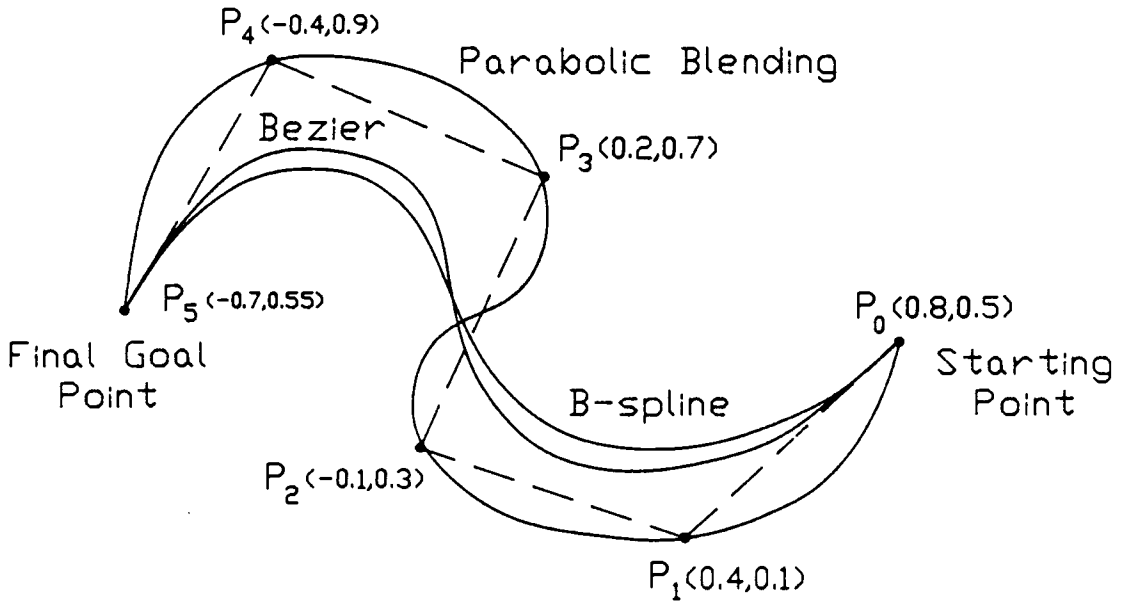
APPENDIX C

CARTESIAN END-EFFECTOR PATHS

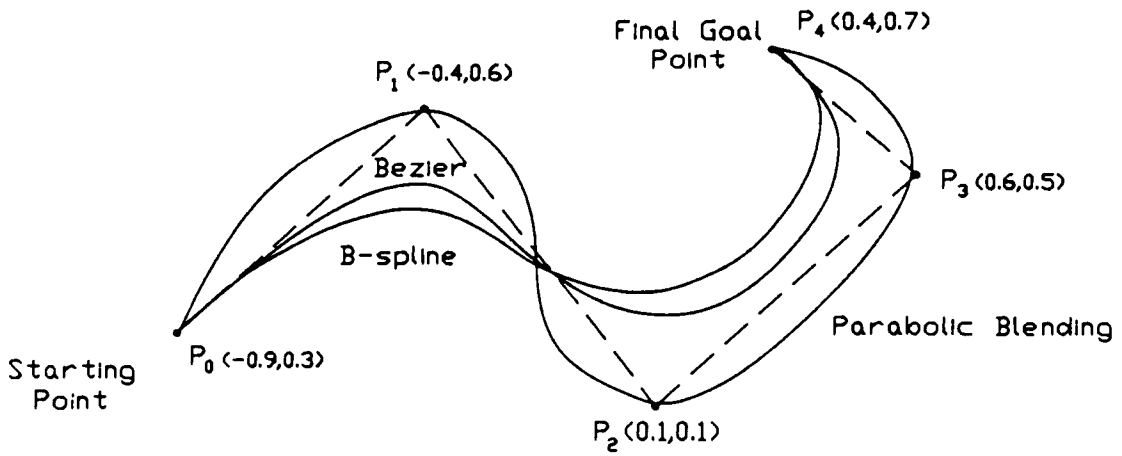




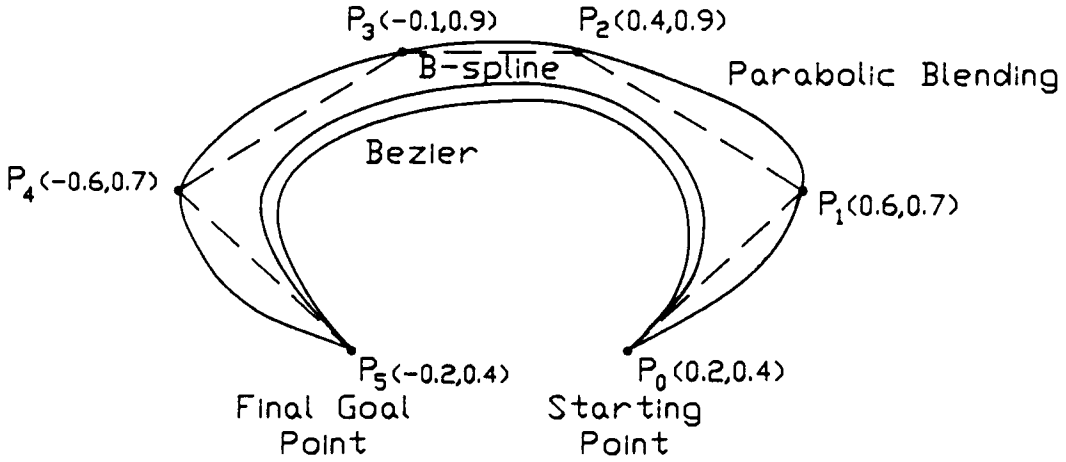
TEST #: 1.3



TEST #: 1.4



TEST #: 1.5



TEST #: 2.1