An Optimum Trajectory Planner for Robot Manipulators in Joint-Space and under Physical Constraints

by:

A.M.S. Zalzala and A.S. Morris

Department of Control Engineering,

University of Sheffield,

Mappin Street,

Sheffield S1 3JD,

U.K.

Research Report # 349

November 1988

Abstract

In the following work, a new trajectory planning algorithm has been developed, where the minimum-time history of the movement of the robot end-effector is defined. Planning is made in the joint-mode by combining a new approach to polynomial splines along with an exhaustive search technique to identify the best minimum-time trajectory. The uniqueness of this algorithm emerges from (1) unique combination of cubic and quadratic polynomials, and (2) the ability to perform the search on local parts of each joint trajectory. A scaling process is applied to these local segments to ensure maximum performance. The method proposed considers all physical and dynamical limitations inherent in the manipulator design, in addition to any geometric constraints imposed on the path. Thus a significant contribution is made through selecting the most accurate, minimum-time, high performance trajectory for a robot. Simulation programs has been written for a study case, and results are reported for a PUMA 600 robot manipulato

Contents

I. Introduction

The Minimum-Time Trajectory Planning problem (MTTP) of robot manipulators is of increasing importance. Recent developments in robotics applications require precise and swift motion to accomplish a specific task. A task is generally defined as the movement of the robot end-effector from a start point to a pre-defined final point. However, it is obvious that an infinite number of solutions exists. Therefore, selecting the minimum-time path among a set of possible solutions is an objective. Such a path would yield better utilization of the capabilities of the manipulator, and leads to a more satisfactory completion of the task.

Over the last few years, research literature has been full with modest contributions in an attempt to solve for the MTTP problem. Such a problem owes its extreme difficulty to several facts, one of which is the nonlinear, coupled nature of the arm dynamics [Paul 1981] , while another is the inherent limitations on the joints actuators. In addition, the presence of any obstacles in the robot work space would introduce the additional significant burden of searching for the best path for the end-effector to traverse. Due to all these difficulties, the solution for the Optimal Control of robot manipulators is divided into two approaches, namely : trajectory planning and trajectory tracking. The intention of this work is directed towards the former, which is usually performed off-line. A trajectory Planner (TP) has the job of describing specific manipulator motion, and generating a suitable space curve (i.e. path) for the robot end-effector to traverse. A time history of positions, velocities, accelerations and torques should be supplied to drive the control loops on the robot joints. Hence, accurate modelling of such a TP is required to maintain correct motion.

Trajectory planning is performed in either Cartesian (World) coordinates or Joint coordinates. A path is usually more realizable in cartesian coordinates, where it is expressed as a set of positions and orientations for the robot end-effector [Lee et al 1986,Fu et al 1987]. In contrast, torques (forces) bounds on the actuators are expressed in joint coordinates. Therefore, one can either transform the given set of the tip locations to joint coordinates, and then solve for the joint-path planning, or alternatively transform the actuators bounds to cartesian coordinates and solve for the cartesian-path planning [Paul 1979]. The latter was investigated in [Paul 1979,Taylor 1977 & 1979,Luh 1981] , and was shown to involve a heavy computational burden [Luh et al 1980]. On the other hand, planning in the joint mode would only involve the transformation of cartesian positions and orientations to its equivalent joint values. Therefore, this type of planning has been investigated extensively [Lee et al 1986]. To achieve such a planning scheme, the cartesian end-effector path is transformed to its equivalent joint-space paths by means of the Jacobian Transformation [Orin and Schrader 1984]. However, since such a transformation is defined only as pointwise , selected points on the cartesian path should be chosen. Hence, apart from the start and end points, several via-points are required. The term *via–points* is widely adopted in research literature, and is adequate for our purposes. These via-points can vary in both number and separation, depending on the path followed. Thus, a longer path may require a larger number of points. In addition, certain geometric constraints may be imposed on the path as in the case of the presence of obstacles, for which more closely-separated via-points are needed.

Although only a few points would be transformed to joint space, the history of a joint trajectory should be continuous in time. To answer for this need, each set of transformed points of each joint is to be fitted by a function. Several methods of curve-fitting were approached. In the case of a set of n via-points, fitting a single $n{-}1$ polynomial is one approach, which is generally accomplished using the Lagrange interpolating polynomial [Burden at al 1974]. Such a polynomial was adopted for the interpolation of robot joint paths in [Finkel 1976]. However, this method has disadvantages in terms of overshooting and wandering [Paul 1972,Lewis 1974,Press et al 1986]. Another approach is by employing polynomial splines, where a low-order polynomial is fitted for each interval between successive via-points. The fitting is made to be smooth by setting the derivatives of adjacent polynomials to the same value at their connecting points. Among all, cubic splines are considered the most common [de Boor 1978,Vandergraft 1978]. These splines are found to be less prone to oscillations because they are limited to third order variations. This approach was widely adopted, and can be found extensively in the literature [Lin et al 1983,Lin and Chang 1985,Jeon and Eslami 1986]. In addition, quadratic splines [Khalil 1984,Castain and Paul 1984] , B-splines [Thompson and Patel 1987] , and least-square approximations [Luh and Lin 1984] have been suggested.

Although several approximation techniques were used, emphasis has been on the planning and minimizing of a single pre-defined path. However, such a path cannot be guaranteed to be the best choice. Several attempts have been made to perform a search sequence in order to identify the best trajectory in the sense of having the minimum traversal time, while conforming to any arm-design limitations. In [Lin et al 1983] , an algorithm was developed to minimize travelling time of a trajectory, employing a direct search method. Also, [Sahar and Hollerbach 1986] introduced a state-space search method created by the tesselation of the path points. Both methods, particularly the second, are computationaly heavy. Similar attempts can be found in

[Rajan 1985,McKay and Shin 1986,Ozaki et al 1987]. However, although having different approaches to all previous algorithms, minimization was made as a global process for all joints, and over the entire trajectory. Such an implementation would utilize the full capabilities of one joint while severely under-utilizing another. One idea for the local-path planning was introduced independently in [Kim and Shin 1985].

In the following work, a new attempt has been made for defining the minimum-time trajectory for a robot manipulator. A unique combination of cubic and quadratic polynomial splines is introduced for the planning of each trajectory in the joint-space. The algorithm, as described, is highly structured, allowing for the maintenance of high manipulator performance for selected repetitive segments of each joint trajectory. As a result, greater time reduction can be achieved. In addition, such a structure allows the use of a highly effective search technique which consequently leads to a more significant travelling-time minimization. The proposed algorithm considers all physical limitations related to the manipulator design, namely: the joints limits on position, velocity, acceleration and jerk. In addition, the dynamics equation of motion [Paul 1981] was considered to ensure the absence of any violations in the torque limitations on the actuators.

The proposed work is presented as follows: In section II, a statement of the problem is defined, while the algorithm is fully illustrated in several parts, in sections III. The significance of the proposed method is discussed in section IV, and simulations and case study results are included in section V. Finally, conclusions are drawn in section VI.

II. Problem Statement

The aim of this work is to construct the optimum trajectories for a mechanical manipulators in joint space. Thus, the main issue under consideration would be the best minimization of travelling time. This class of planning problems was discussed in [Luh and Lin 1981,Brady et al 1982] , where the aspect of fine motion [Kim and Shin 1985] is less significant compared to gaining optimality in the travelling time.

The input to the planner would be N sets of n points, where n is the number of via-points and N is the number of degrees-of-freedom of a given manipulator. In addition, time intervals between each successive points should be supplied by a task planner [Brady et al 1982,Snyder 1985] , which gives an estimation for the total travelling time. Thus, the following must be supplied:

$$\theta_{ij}\ , \quad i=1,2,...,n, \quad j=1,2,...,N \tag{1}$$

$$h_i\ , \quad i=1,2,...,n-1 \tag{2}$$

where h indicates the time intervals.

The set of values in (1) are provided by considering a set of positions and orientations of the robot arm, where each element of such a set is defined as:

$$\mathbf{H}(t_i) = \begin{bmatrix} \mathbf{n}(t_i) & \mathbf{s}(t_i) & \mathbf{a}(t_i) & \mathbf{p}(t_i) \\ 0 & 0 & 0 & 0 \end{bmatrix} \tag{3}$$

where,

$$\mathbf{n}(t_i) = \textit{unit normal vector},$$

$$\mathbf{s}(t_i) = \textit{unit slide vector},$$

$$\mathbf{a}(t_i) = \textit{approach vector}, \text{ and}$$

$$\mathbf{p}(t_i) = \textit{position vector}.$$

Hence, applying an Inverse Kinematics algorithm to each position matrix, $\mathbf{H}(t_i)$, would yield a set of n-length vectors, one for each degee-of-freedom, as expressed in (1). Different inverse kinematics schemes are refered to in [Lee and Ziegler 1984,Paul et al 1981,Bazerghi et al 1984].

For a planning procedure to be accurate, it should take into consideration all realistic constraints which may limit the manipulator performance [Brady et al 1982]. The constraints to be considered in this work are the joint's limits on angular position, velocity, acceleration and jerk. The first of these is imposed by the manipulator geometric design, while the second and third limits are defined by the capabilities of the arm actuators. The limit on jerk is imposed to prevent wear and tear in the actuators [Chand and Doty 1985,Lin et al 1983,Brady et al 1982] , which should yield a longer life cycle. Thus, the following limitations are set:

$$\theta limit_j^- \leq \theta_j(t_i) \leq \theta limit_j^+ \tag{4}$$

$$\dot{\theta}_j(t_i) \leq | \dot{\theta} limit_j | \tag{5}$$

$$\ddot{\theta}_j(t_i) \leq | \ddot{\theta} limit_j | \tag{6}$$

$$\dddot{\theta}_j(t_i) \leq | \dddot{\theta} limit_j | \tag{7}$$

where,

$$\theta limit_j^- = \textit{lower bound on position},$$

$$\theta limit_j^+ = \textit{upper bound on position},$$

$$\dot{\theta} limit_j = \textit{bound on velocity},$$

$$\ddot{\theta} limit_j = \textit{bound on acceleration},$$

$$\dddot{\theta} limit_j = \textit{bound on jerk}.$$

and, $j \in \{1,2,...,N\}$.

In practice, violations of any of these constraints may cause considerable deviations from the planned path.

Another major constraint to be considered is the dynamic equations of motion. These equations have the characteristics of being coupled and highly nonlinear [Luh et al 1980] , and are expressed in terms of positions, velocities and accelerations of each joint. Thus, whilst these limits might appear to be satisfied when the trajectory is defined using static equations, dynamic effects may cause a violation in the value of the allowable torque limit for a joint. The limit on the torque values is given as:

$$\tau_j(t_i) \leq | \tau limit_j | \tag{8}$$

for the same range of i and j.

Including the dynamics equations of motion in the planning of manipulator trajectories was approached by [Lin and Chang 1985,Sahar and Hollerbach 1986]. An algorithm was independantly visualized in [Hollerbach 1984] for the time-scaling of trajectories. This aspect will be discussed in section III.7 following.

The interpolating scheme for this work is chosen to be the Cubic Splines, as they have attractive characteristics. Such a scheme is well known and has its deep roots in mathematical literature [de Boor 1978,Hildebrand 1974,Vandergraft 1978]. Special formulations of cubic splines are required for this application, as discussed in section III.5.

III. The OCQS Algorithm

The algorithm proposed in this work is termed *the Optimum Cubic-Quadratic Splines* , or the *OCQS* for short. This naming will be referred to hereafter.

The OCQS Algorithm will be illustrated over the following sections as follows:

III.1 : Splines Formulations.

III.2 : Time Scaling.

III.3 : Interval Contraction.

III.4 : Time Compensation.

III.5 : Quadratic Approximations.

III.6 : The Search Technique.

III.7 : Dynamics Considerations.

Since almost all stages of the algorithm can be applied to all N joints simultaniously, the procedure will be illustrated for only one joint as far as possible. An indication will be made when it is necessary to consider other joints.

III.1 **Splines Formulations :**

To meet the requirements of the algorithm, the number of via-points n is increased to

$$m = 3n \tag{9}$$

yielding $m-1$ subintervals. This is achieved by subdividing each of the original $n-1$ intervals into *three* equal subintervals. Thus, for a time interval $[t_i\ ,\ t_{i+1}]$ the following is defined:

$$\theta(t_1) = \theta(t_i) + \frac{\theta(t_{i+1}) - \theta(t_i)}{3} \tag{10}$$

$$\theta(t_2) = \theta(t_{i+1}) - \frac{\theta(t_{i+1}) - \theta(t_i)}{3} \tag{11}$$

$$t_1 = t_i + \frac{t_{i+1} - t_i}{3} \tag{12}$$

$$t_2 = t_{i+1} - \frac{t_{i+1} - t_i}{3} \tag{13}$$

An exception is made for intervals 1 and $n-1$, which are divided into *four* equal subintervals, in the same manner. As a result, of these subdivisions, a total of m

points would be available. The initial aim of creating such a construction is to provide enough intervals to fit *cubic polynomials* and *quadratic polynomials* together in the following manner : within each group of three intervals, two successive intervals would be expressed by cubics and the next by a quadratic. Thus, the path would eventually be of the form *(Cubic - Cubic - Quadratic - Cubic - Cubic - ... - Quadratic - Cubic - Cubic).* The path structure is illustrated in figure(1) (segment A), where the *thickened adjacent intervals* are to be expressed by *cubic polynomials* , while other *intermediate intervals* are to be expressed by *quadratic polynomials.* As a result, continuity of second derivatives between cubics and quadratics is to be waived, and only continuity in velocity is to be accounted for. This will be discussed in section IV.

Initially, a cubic spline is set through the *m* points, where boundary conditions for velocity and acceleration would be accounted for, as well as continuity of the first and second derivatives at mid-points. The linear system of equations involved was derived in [Lin et al 1983] , and is included in *Appendix (A)* for reference.

III.2 Time Scaling :

For each pair of adjacent cubic intervals, a scaling procedure is performed to achieve two purposes: (a) Ensure the velocity, acceleration and jerk of both intervals do not exceed certain limits specified by the manipulator design for each of its joints, and (b) Pull up the performance of the manipulator to a maximum, while taking (a) into account.

Thus, for the sample segment B in figure(1), the following is calculated :

$$K_1 = MAX\left[\underset{t \in [t_{i-1},t_i]}{MAX} \left(\frac{\dot{\theta}_{i-1}(t)}{\dot{\theta}limit_j} \right), \underset{t \in [t_i,t_{i+1}]}{MAX} \left(\frac{\dot{\theta}_i(t)}{\dot{\theta}limit_j} \right) \right] \tag{14}$$

$$K_2 = MAX\left[\frac{\ddot{\theta}_{i-1}(t_{i-1})}{\ddot{\theta}limit_j}, \frac{\ddot{\theta}_i(t_i)}{\ddot{\theta}limit_j}, \frac{\ddot{\theta}_i(t_{i+1})}{\ddot{\theta}limit_j} \right] \tag{15}$$

$$K_3 = MAX\left[\frac{\dddot{\theta}_{i-1}(t_{i-1})}{\dddot{\theta}limit_j}, \frac{\dddot{\theta}_i(t_i)}{\dddot{\theta}limit_j} \right] \tag{16}$$

and,

$$K = MAX\left[K_1\ ,\ \sqrt{K_2}\ ,\sqrt[3]{K_3}\right] \tag{17}$$

where, $\dot{\theta}$, $\ddot{\theta}$ and $\dddot{\theta}$ denote velocity, acceleration and jerk, respectively, and K denotes the *Scaling Factor*.

Applying the factor K to the two-interval segment B of figure(1) yields

$$h_{i-1} = K\ .\ h_{i-1} \tag{18}$$

$$h_i = K\ .\ h_i \tag{19}$$

$$\dot{\theta}_{i-1}(t_{i-1}) = \frac{1}{K}\ .\ \dot{\theta}_{i-1}(t_{i-1}) \tag{20}$$

$$\dot{\theta}_i(t_i) = \frac{1}{K}\ .\ \dot{\theta}_i(t_i) \tag{21}$$

$$\dot{\theta}_i(t_{i+1}) = \frac{1}{K}\ .\ \dot{\theta}_i(t_{i+1}) \tag{22}$$

$$\ddot{\theta}_{i-1}(t_{i-1}) = \frac{1}{K^2}\ .\ \ddot{\theta}_{i-1}(t_{i-1}) \tag{23}$$

$$\ddot{\theta}_i(t_i) = \frac{1}{K^2}\ .\ \ddot{\theta}_i(t_i) \tag{24}$$

$$\ddot{\theta}_i(t_{i+1}) = \frac{1}{K^2}\ .\ \ddot{\theta}_i(t_{i+1}) \tag{25}$$

$$\dddot{\theta}_{i-1}(t_{i-1}) = \frac{1}{K^3}\ .\ \dddot{\theta}_{i-1}(t_{i-1}) \tag{26}$$

$$\dddot{\theta}_i(t_i) = \frac{1}{K^3}\ .\ \dddot{\theta}_i(t_i) \tag{27}$$

where the marker (.) denotes scalar multiplication.

The scaling procedure can be verified by considering

$$\theta_{i+1} = \theta_i + \dot{\theta}_i\ .\ h_i + \ddot{\theta}_i\ .\ h_i^2 + \dddot{\theta}_i\ .\ h_i^3 \tag{28}$$

then, multiplying h_i by a factor K would cause $\dot{\theta}$, $\ddot{\theta}$ and $\dddot{\theta}$ to be multiplied by $\frac{1}{K}$, $\frac{1}{K^2}$ and $\frac{1}{K^3}$, respectively. This method not only assures that joint performance, for this particular segment, is kept within limits, but also that it is pulled up to a maximum, yielding a minimum of time.

III.3 Interval Contraction :

For each interval in the two-cubic segment, location of maximum velocity is found, and the corresponding position cubic function is truncated accordingly. Thus, for a time interval $[t_i,t_{i+1}]$, the maximum velocity could be at t_i , t_{i+1} or $t \in [t_i,t_{i+1}]$, which would cause the corresponding time interval to vanish, remain as it is, or be truncated at some value, respectively, in accordance with the standard properties of a cubic equation.

III.4 Time Compensation :

At this stage, as all segments of two-cubics are optimised, they are to be connected by approximated quadratic segments similar to that shown in segment C of figure(1). Values of the first derivative (i.e. velocity) at the end points are already specified by the neighbouring cubics. Since the second derivative of a quadratic is a constant, each quadratic interval is initially set so as to get the maximum allowable acceleration $\ddot{\theta}limit_j$,

$$h_{quad} = \left| \frac{\dot{\theta}_{i+2}(t_{i+3}) - \dot{\theta}_{i+2}(t_{i+2})}{\ddot{\theta}limit_j} \right| \tag{29}$$

Hence, as a result of this approximation, corresponding intervals of each of the N joint-paths will have different values, leading to a different total travelling time between *start* and *end* points. This situation is best simulated by figure(2), assuming a 4 via-points path, (i.e. n=4, m=12), and 6 joints (i.e. N=6). Thickened lines represent cubic equations, while light lines represent quadratics. Obviously, since each joint is being treated individually, a different total travelling time occurs for each one. One joint is expected to reach the end point faster than the others (e.g. joint 4 in figure(2)). Thus, it is appropriate to add some additional time

$$t_j^{add} \quad , j=1,2,...,N \tag{30}$$

for each lagging joint to make its total travelling time equal to that of the leading joint. In figure(2), t_j^{add} time is added for $j \in \{1,2,3,5,6\}$ to achieve equal travelling time for all six joints.

However, t_j^{add} can be added in two ways. It could be added so as to maintain equal time between each successive pair of original via-points, i.e. the n points given by the inverse kinematics transformation. This could be undertaken when path constrains are imposed, and the end-effecter is to be forced to go exactly through certain positions during movement. Alternatively, t_j^{add} could be added in a manner where no constraints are put on the end-effector's path, and all that is needed is to move it between start and end points. It should be mentioned that any time addition should be made to the quadratic intervals only. As for the minimization of time, the first approach would cost more, since certain restrictions were made on the path.

The two methods of adding t_j^{add} will be discussed now; a quadratic interval will be refered to as h_{quad}.

III.4.1 *Time Addition – Constrained path* :

The value of h_{quad} in each of the $n-1$ segments of each joint is adjusted so as to satisfy (a) The maximum allowable acceleration (*eqn.*29), and (b) Equality of total time of each segment over all joints. This would maintain via-point constraints. The value of the added time would be

$$t_j^{add} = \sum_{i=1}^{no.\ of\ quadratics} \left[\underset{j=1,2,..,N}{MAX} \left(h_{quad}^{i,j} \right) - h_{quad}^{i,j} \right] \tag{31}$$

Now, to fit in a quadratic equation between any two positions $\theta_i(t_i)$ and $\theta_i(t_{i+1})$, and velocities $\dot{\theta}_i(t_i)$ and $\dot{\theta}_i(t_{i+1})$, the corresponding time interval must be given as

$$h_{quad}^* = 2\left[\frac{\theta_i(t_{i+1}) - \theta_i(t_i)}{\dot{\theta}_i(t_i) + \dot{\theta}_i(t_{i+1})} \right] \tag{32}$$

However, since h_{quad} cannot be guaranteed to equal h_{quad}^*, an approximation for the quadratic polynomial must be made. The derivation of (eqn.32) can be found in *Appendix (B)*.

III.4.2 *Time Addition – Unconstrained path* :

It is realized that for a quadratic polynomial to fit correctly, its time interval must be h_{quad}^* of (eqn.32), while the initially assumed interval is h_{quad}. Thus a difference of

$$h_{diff} = h_{quad}^* - h_{quad} \tag{33}$$

must be compensated for. And for the whole path,

$$h_{add} = \sum_{i=1}^{no.\ of\ quadratics} h_{diff,i} \tag{34}$$

must be added.

However, since total compensation time was predefined as t_j^{add}, (eqn.30), the quadratic interval of each segment i is set as

$$h_{quad,i} = \left[\frac{h_{diff,i}}{h_{add}}\right] \cdot t_j^{add} \quad , i = 1,2,\ldots, \textit{no. of quadratics} \tag{35}$$

where t_j^{add} is divided proportionally between all quadratic segments in the path, depending on the value of h_{diff} of each. However, since the computed h_{quad} cannot be guarenteed to equal h^*_{quad}, an approximation for the equation is again needed.

As a result of applying the procedure of section III.4.2 on the example of figure(2), equal time trajectories would be expressed as in figure(3).

III.5 Quadratic Approximations :

An approximation must be made for the quadratic polynomial between positions $\theta_i(t_i)$ and $\theta_i(t_{i+1})$, and velocities $\dot{\theta}_i(t_i)$ and $\dot{\theta}_i(t_{i+1})$, in a period h_{quad}. The approximation is accomplished by utilizing an equal interval, large-number-of-segments cubic spline. The time interval h_{quad} is divided into l equal sub-intervals

$$h_{div} = \frac{h_{quad}}{l} \tag{36}$$

Since the velocity during interval h_{quad} is a first order polynomial (i.e. a straight line), the corresponding velocity at the start of each h_{div} interval is calculated as

$$\dot{\theta}_w = \dot{\theta}_i(t_i) + w \,.\, \left[\frac{\dot{\theta}_i(t_{i+1}) - \dot{\theta}_i(t_i)}{l}\right] \;, \qquad w = 1,2,\ldots,l-1 \tag{37}$$

Thus a system of $l-2$ equations in $l-2$ *position* variables is constructed, taking into account values of h_{div} and $\dot{\theta}_w$ calculated. The system is represented as

$$\mathbf{A} \,.\, \mathbf{x} = \mathbf{b} \tag{38}$$

where,

$$\mathbf{A} = \begin{bmatrix} \frac{-6}{h_{div}^2} & \frac{3}{h_{div}^2} & 0 & . & . & . & . & 0 \\ 0 & \frac{3}{h_{div}^2} & \frac{-6}{h_{div}^2} & \frac{3}{h_{div}^2} & . & . & . & . \\ . & 0 & . & . & . & . & . & . \\ . & . & . & . & . & . & . & . \\ . & . & . & . & . & . & . & 0 \\ 0 & . & . & . & . & 0 & \frac{3}{h_{div}^2} & \frac{-6}{h_{div}^2} \end{bmatrix} \tag{39}$$

and,

$$\mathbf{B} = \begin{bmatrix} \dfrac{-3\theta_1}{h_{div}^2} \\ \dfrac{-\dot{\theta}_1 - \dot{\theta}_2 + 2\,\dot{\theta}_3}{h_{div}} \\ \cdot \\ \cdot \\ \cdot \\ \dfrac{-3\theta_l}{h_{div}^2} \end{bmatrix} \tag{40}$$

and $\mathbf{x}$ is the required positions vector,

$$\mathbf{x} = \begin{bmatrix} \theta_2 \\ \theta_3 \\ \cdot \\ \cdot \\ \theta_{l-2} \\ \theta_{l-1} \end{bmatrix} \tag{41}$$

It was found that as $l \to \infty$, then $h_{div} \to 0$, and the cubic polynomials concerned tend to form a quadratic. However, the value of l should be properly chosen for each approximation, to assure minimum discontinuity among sub-intervals, and to avoid unnecessary computational burden.

The linear system described has been derived utilizing the *Hermit interpolation polynomial*. Detailed derivations are included in *Appendix (C)*.

III.6 The Search Technique :

Up to this stage, planning procedure has been concerned with a single pre-defined path. However, the trajectory produced is not necessarily the best choice. Therefore, a search procedure has to be employed to obtain optimality in terms of the travelling time.

The search is performed by varying the positions of the end-points of each two-cubic segment of a joint-path, figure(1)-B, by a certain value, θ_{var}. The variation is done on the initial via-points of (*eqn.*9), and the spline fitting of section III.1 is applied yielding a *new* trajectory. This new trajectory is then processed by the *scaling* and *contraction* procedures of sections III.2 and III.3, yielding the minimum of time for each two-cubic segment. The segments are then compared to their equivalents in the original trajectory, according to the following:

Refering to figure(1), considering a two-cubic segment B with its middle point at position $\theta_i(t_i)$, we define

$$t_{value}^{orig} = h_i^{orig} + h_{i-1}^{orig} + \frac{\dot{\theta}^{orig}(t_{i-1}) - \dot{\theta}^{orig}(t_{i-2})}{\ddot{\theta}limit_j} \tag{42}$$

and,

$$t_{value}^{new} = h_i^{new} + h_{i-1}^{new} + \frac{\dot{\theta}^{new}(t_{i-1}) - \dot{\theta}^{orig}(t_{i-2})}{\ddot{\theta}limit_j} \tag{43}$$

where superscripts *orig* and *new* denotes *original* and *new* trajectories, respectively. Thus, for a new segment to be accepted, the following condition must be satisfied:

$$t_{value}^{new} < t_{value}^{orig} \tag{44}$$

This condition should be satisfied for all cubic segments in the trajectory, except for the first, for which the following condition is imposed

$$h_i^{new} + h_{i-1}^{new} < h_i^{orig} + h_{i-1}^{orig} \tag{45}$$

Thus, some of the two-cubic segments of the original trajectory will be replaced by new less time-consuming segments from the new trajectory, and new corresponding quadratics are fitted, leading to a further reduction in traverse time.

The previous procedure is then repeated with a negative added value to end-points positions, namely $-\theta_{div}$, and a yet newer trajectory is constructed, where the replacement process is carried out under the same conditions. The situation so far is shown in figure(4) for an 8-via-points joint-path. It should be mentioned that the original n via-points are not altered during the search, but rather kept stationary.

Now, the value of θ_{div} is to be increased to $2\theta_{div}$, and *two* other new trajectories are constructed and processed in a similar manner.

This task is to be continued for

$$\theta_{add}^{u} = u \,.\, \theta_{div} \qquad , \; u = \pm\, 1,2,3,... \tag{46}$$

until no more two-cubic segments could be exchanged.

However, it should be emphasized that the value of θ_{add}^{u} must not exceed the position bounds imposed by (*eqn*.4). If such a situation occurs, θ_{add}^{u} should be set to the corresponding bound value.

The initial task of the search process is to vary the value of velocity for each two-cubic segment, leading to a smaller scaling factor, K, and therefore less time.

The value of θ_{div} is chosen to be relatively large at the start, (e.g. 10 *degrees*), which was found to give better results than a smaller value. This will be illustrated in section V.

The search is stopped once the total trajectory time for a path with an incremental value $\theta_{add}^{u_{i+1}}$ is equal to that with a value of $\theta_{add}^{u_i}$, where $u_i \in \{1,2,3,...\}$ and $u_{i+1} \in \{2,3,...\}$.

Nevertheless, since a large incremental value of θ_{div} was used, larger minimization could be achieved considering a value $\theta_{add}^{u_i^*}$, where

$$\theta_{add}^{u_i} < \theta_{add}^{u_i^*} < \theta_{add}^{u_{i+1}} \tag{47}$$

Thus, a new search phase is initiated, with a start position of $\theta_{add}^{u_i}$ and an incremental value of

$$\theta_{add}^{u} = u \, . \, \frac{\theta_{div}}{10} \quad , \; u = \pm\, 1,2,3,... \tag{48}$$

where a new smaller travelling time would be achieved.

A difference tolerance is assumed,

$$tol = t_{total}^{1} - t_{total}^{2} \tag{49}$$

which represents the difference in the total travelling time between one phase of the search and the proceeding phase. If the value of *tol* was not met by the first two search phases described, then a third phase is to be initiated with

$$\theta_{add}^{u} = u \, . \, \frac{\theta_{div}}{100} \quad , \; u = \pm\, 1,2,3,... \tag{50}$$

and so forth until the value of *tol* is met.

Once the search is completed, the time compensation process of section III.5 is applied to the optimum trajectories obtained for all joints.

III.7 Dynamics Considerations :

The limits on the trajectory performance considered in (*eqns*.4–7) are constant approximate values that were deduced for simplicity [Paul 1979,Hollerbach 1984]. However, since the presence of interaction between different manipulator joints is a fact, such an approximation is not totally reliable [Hollerbach 1984]. Attempts were made for the determination of exact time-dependent values for these limits by [Vukobratovic and Kircanski 1982,Bobrow et al 1985]. Unfortunately, the proposed methods proved to be computationally unacceptable, which emphasized the need for the development of a time-scaling procedure of moderate computational burden. One algorithm was independently developed by [Hollerbach 1984], while another algorithm was introduced by [Lin and Chang 1985]. Both proposed methods proceed in the direction of computing the modified trajectory's dynamics by means of linear combination of the original dynamics values.

The equations of motion for a manipulator is defined by [Paul 1981,Fu et al 1987] as

$$\tau_j = \sum_{q=1}^{N} D_{jq}\ddot{\theta}_q + \sum_{q=1}^{N}\sum_{m=1}^{N} h_{jqm}\,\dot{\theta}_q\,\dot{\theta}_m + c_j \qquad (51)$$

$$, \; j = 1,2,\ldots,N$$

or, alternatively, in matrix form as

$$\boldsymbol{\tau}(t) = \mathbf{D}(\boldsymbol{\theta}(t))\,\ddot{\boldsymbol{\theta}}(t) + \mathbf{h}(\boldsymbol{\theta}(t),\dot{\boldsymbol{\theta}}(t)) + \mathbf{c}(\boldsymbol{\theta}(t)) \qquad (52)$$

where,

$$\boldsymbol{\tau}(t) = [\,\tau_1(t), \tau_2(t), \cdots, \tau_N(t)]^T = N\times1 \textit{ generalized torque vector },$$

$$\boldsymbol{\theta}(t) = [\,\theta_1(t), \theta_2(t), \cdots, \theta_N(t)]^T = N\times1 \textit{ vector of joint variables },$$

$$\dot{\boldsymbol{\theta}}(t) = [\,\dot{\theta}_1(t), \dot{\theta}_2(t), \cdots, \dot{\theta}_N(t)]^T = N\times1 \textit{ vector of joint velocity },$$

$$\ddot{\boldsymbol{\theta}}(t) = [\,\ddot{\theta}_1(t), \ddot{\theta}_2(t), \cdots, \ddot{\theta}_N(t)]^T = N\times1 \textit{ vector of joint acceleration },$$

$$\mathbf{D}(\boldsymbol{\theta}(t)) = N\times N \textit{ inertial acceleration-- related matrix },$$

$$\mathbf{h}(\boldsymbol{\theta}(t),\dot{\boldsymbol{\theta}}(t)) = N\times1 \textit{ nonlinear coriolis} \text{ and } \textit{centrifugal force vector}, \text{ and}$$

$$\mathbf{c}(\boldsymbol{\theta}(t)) = N\times1 \textit{ gravity force vector }.$$

Thus, for a new constructed trajectory segment, and assuming a scale factor, K^*, the following is deduced for time t_i,

$$\dot{\theta}_i^{new} = \frac{\dot{\theta}_i}{K^*}$$

$$\ddot{\theta}_i^{new} = \frac{\ddot{\theta}_i}{(K^*)^2}$$

substituting these values in (*eqn.*51) gives :

$$\tau_j^{new} = \sum_{q=1}^{N} D_{jq} \frac{\ddot{\theta}_q}{(K^*)^2} + \sum_{q=1}^{N} \sum_{m=1}^{N} h_{jqm} \frac{\dot{\theta}_q \, \dot{\theta}_m}{(K^*)^2} + c_j \tag{53}$$

Then simplifying yields :

$$\tau_j^{new} = \frac{1}{(K^*)^2} \left[\tau_j - c_j\right] + c_j \tag{54}$$

Now, considering the limit on torque imposed by (*eqn*.8), then,

$$\left[\frac{1}{(K^*)^2} \left[\tau_j - c_j\right]\right] + c_j \leq |\ \tau limit_j\ | \tag{55}$$

or,

$$K^* = \sqrt{\frac{\tau_j - c_j}{|\ \tau limit_j\ | - c_j}} \tag{56}$$

The minimum-time joint-paths constructed by the *OCQS* algorithm are to be tested to maintain limits on torques. However, since a limit was considered for both velocity and acceleration values (i.e. *eqns*. 5 and 6), the scaling factor K^* must satisfy

$$K^* \geq 1 \tag{57}$$

to eliminate any possible overloading in velocity or acceleration.

Thus, the role of the dynamic scaling procedure is to work as a safeguard against any breach in a torque value for a joint, due to its coupling with others. Consequently, the total travelling time along the path could either *increase* if a joint torque is above limit, or alternatively *maintain* its previously predicted value.

The usual approach for scaling is to get a global value for K^* over the trajectories of all joints, and then scale all N paths simultaneously. However, this is found to be inefficient in terms of time, since one joint capabilities may be under-utilized for the whole path of motion because of a violation in another joint at one particular point. Thus, in our approach to solving such a problem, scaling is once again performed on local selected segments of the constructed trajectories.

Considering the structure of the *OCQS* algorithm, a violation in the torque value of a joint can occur either at a cubic or at a quadratic segment. Hence, scaling must be performed for that segment and the corresponding cubic or quadratic segments of all other joints. The procedure for scaling both types of segments is discussed.

III.7.1 Violation at a Cubic :

For this case, the situation is illustrated in figure (5) for a single joint. Employing the local scaling factor, K^*, new values for h, $\dot{\theta}$, $\ddot{\theta}$ and $\dddot{\theta}$ for intervals $[t_{i-1},t_i]$ and $[t_i,t_{i+1}]$ are scaled using (*eqns.* 18–27). As for the neighbouring quadratic interval $[t_{i-2},t_{i-1}]$, a new value for $h_{quad,i-2}$ is set as

$$h^{new}_{quad,i-2} = \frac{\dot{\theta}^{new}_{i-1}(t_{i-1}) - \dot{\theta}_{i-2}(t_{i-2})}{\ddot{\theta}_{i-2}} \tag{58}$$

where,

$$\ddot{\theta}_{i-2} = \frac{\dot{\theta}_{i-1}(t_{i-1}) - \dot{\theta}_{i-2}(t_{i-2})}{h_{quad,i-2}} \tag{59}$$

Similarly, for interval $[t_{i+1},t_{i+2}]$ we get

$$h^{new}_{quad,i+1} = \frac{\dot{\theta}_{i+1}(t_{i+2}) - \dot{\theta}^{new}_{i+1}(t_{i+1})}{\ddot{\theta}_{i+1}} \tag{60}$$

where,

$$\ddot{\theta}_{i+1} = \frac{\dot{\theta}_{i+1}(t_{i+2}) - \dot{\theta}_{i+1}(t_{i+1})}{h_{quad,i+1}} \tag{61}$$

III.7.2 *Violation at a Quadratic :*

The situation is again simulated in figure (6) for one joint, where the violation is assumed to occur during interval $[t_i,t_{i+1}]$. Thus, h, $\dot{\theta}$, $\ddot{\theta}$ and $\dddot{\theta}$ parameters for the neighbouring cubic intervals $[t_{i-2},t_i]$ and $[t_{i+1},t_{i+3}]$ are scaled by a local factor K^* using (*eqns.* 18–27). Accordingly, a reduction in $\dot{\theta}_i(t_i)$ and $\dot{\theta}_i(t_{i+1})$ by a factor of $\frac{1}{K^*}$ would cause the quadratic acceleration, $\ddot{\theta}_i$, to reduce by the same factor. However, according to (*eqn.* 53), $\ddot{\theta}_i$ is to be reduced by a factor of $\frac{1}{K^{*2}}$. Hence, $h_{quad,i}$ should be adjusted to account for the difference,

$$h^{new}_{quad,i} = \frac{\dot{\theta}_i(t_{i+1}) - \dot{\theta}_i(t_i)}{\left[\frac{\ddot{\theta}_i}{K^{*2}}\right]} \tag{62}$$

where $\ddot{\theta}_i$ is set to its new value,

$$\ddot{\theta}^{new}_i = \frac{\ddot{\theta}_i}{(K^*)^2} \tag{63}$$

The side quadratic intervals $[t_{i-3},t_{i-2}]$ and $[t_{i+3},t_{i+4}]$ are treated as in section III.7.1 previously.

Once the scaling procedure is completed for the violated joint-path, other joint paths are treated similarily employing either the procedure of section III.7.1 or III.7.2 as appropriate.

IV. Discussion

In the previous section, an algorithm for the robot MTTP problem was introduced. It is intended now to discuss and verify the different technical and numerical aspects for each procedure proposed. Throughout the development of this algorithm, two important aspects in the MTTP were given priority: time optimality and path accuracy. Robot manipulators are expected to execute a certain task in a minimum of time. However, during this optimal planning procedure, certain accuracy is expected within the controller. This accuracy may be corrupted by mis-utilizing joint actuators through applying some unrealistic burden on their performance. In the methods of joint-trajectory planning, the initial time required by the manipulator is estimated by a *task planner* [Brady et al 1982]. However, this estimate may force the manipulator to deviate from its pre-planned task if saturation occurs in any of its actuators. The initial calculation of time is tested and correction is applied if any of the three constraints, namely velocity, acceleration or jerk exceeds a specified limit. The values of velocity, acceleration and jerk are kept to their maximum allowable state, to utilize the actuator's capabilities in full. This was found appropriate since several parts of a joint path could be scaled individually, yielding a large reduction in time. As for the effect of maximizing acceleration to its limits on actuator performance, the jerk is set as a safeguard. It is defined as the rate of change of acceleration [Brady et al 1982,Lin et al 1983], and is expected to protect against wear and tear if kept within a certain limit. It was realised that maintaining high velocity movement would cause a considerable reduction in travelling time. Therefore, location of maximum velocity in each cubic interval was found. To achieve minimum travelling time between these maximum values of velocity, linear variation should be implemented. A first order equation was used for velocity, giving a quadratic equation for position. However, this assumption caused a constant acceleration value, which waived away the idea of acceleration continuity. The idea of using both *second* and *first* order polynomials for velocity was adopted to maintain smoothness in transitions throughout the path. The use of a *first* order polynomial throughout motion may cause considerable roughness, since sudden changes in velocity are to be expected at transition points (i.e. via-points). In such a situation, a manipulator would be caused to stop at these points [Khalil 1984].

During the performance of a task, certain constraints may be imposed on the required path, such as passing the end-effecter through specific locations. When this is the case, the given via-points must be available on the new planned path. If not, the problem is reduced to the simple matter of moving the end-effector from one point in space to another. This is expected to be usually faster, since there are no constraints in terms of special passage points. The proposed algorithm accounts for both cases.

When the required time for a *quadratic* is calculated, a set of splined functions has been adopted as an approximation. Since the difference between calculated quadratic intervals and real interval values changes from one segment to another, the number of splined functions would change correspondingly. A greater difference is expected to lead to a larger set of splines.

As for the search technique suggested, the variation in the end-point positions of the two-cubic segments is governed by the position bounds on each joint, (*eqn*.4). Thus, varying these bounds for each segment throughout the path would be of great help if obstacles are expected in the work area, where the path would be restricted locally within certain bounds.

Finally, it should be outlined that the *OCQS* algorithm is most applicable to the planning of long paths, with a relatively large number of via-points, where the main goal is to minimize the execution time. This is quite different from the concept of *fine motion* , for which the manipulator is expected to make short moves, and where there are many other important factors apart from travelling time. With this idea kept in mind, the discontinuity in the acceleration profile of the planned trajectory becomes acceptable [Luh and Lin 1981,Kim and Shin 1985]. This type of disadvantage is compensated for generously by the fact of optimum minimization of time, for which gross-motion and fast controls could be achieved.

V. Simulation Results

In this section, results applying the *OCQS* algorithm for the trajectory planning of a *PUMA* 600 robot manipulator, with 6 degrees-of-freedom, are reported. Programs has been written in the *C* programing language on the *Sun Workstations* running under *UNIX*. An 8-via-points path has been chosen as an input, giving 6 sets of 8 joint values, as illustrated in Table(1). Initial values for the time intervals were calculated by a global scaling process [Lin 1983], taking into account the physical limitations on the manipulator.

The limitations for each of the manipulator joints are tabulated in Table (2).

These 8 points are expanded to 24 via-points, as required by the *OCQS* structure, giving a total of 23 intervals. Applying the *scaling* and *contraction* procedures to the trajectory of each joint separately, and then compensating for time differences leads to a reduction in time from (32.481) seconds to (14.215) seconds.

Once the search technique is applied, time was minimized from (32.481) seconds to (7.167) seconds. Since the search was conducted for each joint-trajectory separately, different numbers of search-phases were detected. Hence, while one joint required only *two* phases to reach its optimum trajectory, another required as much as *four* phases. This indicates the great importance of distributing the algorithm on separate local segments of each joint-trajectory. The search process for *joint*#1 is shown in figure (7), indicating the position profile.

The significance of the *contraction* procedure is obvious through the fact that, for all joints, up to *four* cubic intervals vanished when contracted. This, again, greatly owes its success to the search performed.

As for the dynamic equations of motion, the Recursive Lagrangian Formulation introduced by [Hollerbach 1981] was adopted. A full data model for the *PUMA* 600 robot is included in *Appendix (D)*.

The results of this simulation are summarized in Table (3). The constructed optimum joint trajectories are shown for all 6 joints of the robot in figures (8)-(13), where position, velocity, acceleration and jerk profiles are plotted. In addition, torque profiles of joints 1 and 3 are included in figures (14) and (15).

The data used in this study case were obtained from [Lin et al 1983,Thompson and Patel 1987] for comparison purposes, while the *PUMA* model data were extracted from [Leahy et al 1986,Armstrong et al 1986].

Table (1) : Joint Variables						
Point#	Joint#					
	1	2	3	4	5	6
1	10	15	45	5	10	6
2	60	25	180	20	30	40
3	75	30	200	60	-40	80
4	130	-45	120	110	-60	70
5	110	-55	15	20	10	-10
6	100	-70	-10	60	50	10
7	-10	-10	100	-100	-40	30
8	-50	10	50	-30	10	20

Table (2) : Limits						
Limit	Joint#					
	1	2	3	4	5	6
Velocity ($deg.sec^{-1}$)	100	95	100	150	130	110
Acceleration ($deg.sec^{-2}$)	45	40	75	70	90	80
Jerk ($deg.sec^{-3}$)	60	60	55	70	75	70
Torque ($N.m.$)	97.6	186.4	89.4	24.2	20.1	21.3

Table (3) : Trajectories Time-Reduction History

Joint-Trajectory No.	Assumed Time	Initial Trajectory Time	Search Phases*	Searched Paths	Final Minimized Time	Final Compensated Time	Vanished Intervals
1	32.481	8.864	4	40	5.532	7.167	4
2	32.481	7.417	3	12	6.126	7.167	2
3	32.481	12.734	4	60	5.691	7.167	3
4	32.481	12.350	2	12	4.886	7.167	3
5	32.481	10.164	2	10	5.125	7.167	3
6	32.481	7.710	2	8	5.154	7.167	3

* Assuming a difference tolerance of *tol*=0.01

VI. Conclusion

A new algorithm for the minimum-time trajectory planning of robot manipulators in joint space has been developed. Although planning is initiated with a pre-defined trajectory, a search technique is employed to define the optimum trajectory, through the minimization of selected local segments of each separate joint-trajectory. All realistic physical constraints forced by the manipulator design and the surrounding work volume are taken into consideration. In addition to obtaining minimum-time joint-trajectories, one side result that is of great importance is the algorithm's ability to be adopted for obstacle avoidance tasks. The proposed algorithm is very structured in formulation, which suggests its adoption as an on-line scheme for trajectory planning. The feasability of such an idea is currently under investigation.

References

ARMSTRONG, B., KHATIB, O., AND BURDICK, J., (1986). "The Explicit Dynamic Model and Inertial Parameters of the PUMA 560 Arm," in *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 510-18.

BAZERGHI, A., GOLDENBERG, A. A., AND APKARIAN, J., (1984). "An Exact Kinematic Model of PUMA 600 Manipulator," in *IEEE Trans. SMC-14*, pp. 483-87.

BOBROW, J. E., DUBOWSKY, S., AND GIBSON, J. S., (1985). "Time-Optimal Control of Robotic Manipulators Along Specified Paths," in *Int. J. Robotics Research*, vol. 3, pp. 3-17.

BRADY, J. M., HOLLERBACH, J. M., JOHNSON, T. L., LOZANO-PEREZ, T., AND MASON, M. T.(1982), *Robot Motion : Planning and Control*, MIT Press.

BURDEN, R. L., FAIRES, J. D., AND REYNOLDS, A. C.(1974), *Numerical Analysis*, PWS Pub..

CASTAIN, R. H. AND PAUL, R. P., (1984). "An On-Line Dynamic Trajectory Generator," in *Int. J. Robotics Research*, vol. 3, pp. 68-72.

CHAND, S. AND DOTY, K. L., (1985). "On-Line Polynomial Trajectories for Robot Manipulators," in *Int. J. of Robotics Research*, vol. 4, pp. 38-48.

FINKEL, R. A.(1976), *Constructing and Debugging Manipulator Programms*, Ph.D. Dissertation, Stanford University.

FU, K. S., LEE, C. S. G., AND GONZALIS,(1987), *Robotics : Control, Sensing, Vision and intelligence*, McGraw Hill.

HILDEBRAND, F. B.(1974), *Introduction to Numerical Analysis*, McGraw Hill.

HOLLERBACH, J. M., (1981). "A Recursive Formulation of Lagrangian Manipulator Dynamics," in *IEEE Trans. SMC-10*, pp. 730-36.

HOLLERBACH, J. M., (1984). "Dynamic Scaling of Manipulator Trajectories," in *Trans. ASME, J. of Dyn. Syst., Meas. and Control* , pp. 102-06.

JEON, H. T. AND ESLAMI, M., (1986). "On Minimum-Time Joint-Trajectory Planning For Industrial Manipulators With Cubic Polynomial and Input Torque Constraints," in *Proc. IEEE 25th Conf. on Decision and Control*, pp. 435-40.

KHALIL, W., (1984). "Trajectories Calculations in the Joint Space of Robots," in *'Advanced Software in Robotics', A. Danthine and M. Geradin (eds.)*, Elsevier Science Pub. B. V. (North-Holland).

KIM, B. K. AND SHIN, K. G., (1985). "Minimum-Time Path Planning fo Robot Arms and Their Dynamics," in *IEEE Trans. SMC-15*, pp. 213-23.

LEAHY, M. B., JR., NUGENT, L. M., VALAVANIS, K. P., AND SARIDIS, G. N., (1986). "Efficient Dynamics for a PUMA 600," in *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 519-24.

LEE, C.S.G. AND ZIEGLER, M., (1984). "A Geometric Approach in Solving The Inverse Kinematics of PUMA Robot," in *IEEE Trans. AES-20*, pp. 695-706.

LEE, C. S. G., FU, K. S., AND GONZALES, R. C., (1986). "Planning of Manipulator Trajectories," in *'Tutorial on Robotics'*, pp. 155-65.

LEWIS, R. A.(1974), *Autunomous Manipulation on a Robot : Summary of Manipulator Software Functions*, Tech. Memo 33-679, J.P.L., California, U.S.A..

LIN, C. S., CHANG, P. R., AND LUH, J. Y. S., (1983). "Formulation and Optimization of Cubic Polynomial Joint Trajectories For Industrial Robots," in *IEEE Trans. AC-28*, pp. 1066-74.

LIN, C. S. AND CHANG, P. R., (1985). "Approximate Optimum Paths of Robot Manipulators Under Realistic Physical Constraints," in *Proc. IEEE Int. Conf. on Robotics and Automation*, pp. 737-42.

LUH, J. Y. S., WALKER, M. W., AND PAUL, R. P. C., (1980). "On-Line Computational Scheme for Mechanical Manipulators," in *Trans. ASME, J. of Dyn. Syst., Meas. and Control*, vol. 102, pp. 69-76.

LUH, J. Y. S. AND LIN, C. S., (1981). "Optimum Path Planning for Mechanical Manipulators," in *Trans. ASME, J. of Dyn. Syst., Meas. and Control*, vol. 102, pp. 142-51.

LUH, J. Y. S. AND LIN, C. S., (1984). "Approximate Joint Trajectories For Control of Industrial Robots Along Cartesian Paths," in *IEEE Trans SMC-14*, pp. 444-50.

MCKAY, N. D. AND SHIN, K. G., (1986). "Minimum-Time Trajectory Planning for Industrial Robots with General Torque Constraints," in *Proc. 1986 IEEE Int. Conf. on Robotics and Automation*, vol. 1, pp. 412-17.

ORIN, D. E. AND SCHRADER, W. W., (1984). "Efficient Computation of the Jacobian for Robot Manipulators," in *Int. J. Robotics Research*, vol. 3, pp. 66-75.

OZAKI, H., YAMAMOTO, M., AND MOHRI, A., (1987). "Optimal and Near-Optimal Manipulator Joint Trajectories With a Preplanned Path," in *Proc. 26th Conf. on Decision and Control*, pp. 1029-34.

PAUL, R. P.(1972), *Modeling, Trajectory Calculation and Servoing of a Computer Controlled Arm'*, Ph.D. Dissertation, Stanford University.

PAUL, R. P., (1979). "Manipulator Cartesian Path Control," in *IEEE Trans. SMC-9*, pp. 702-11.

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

PAUL, R. P., SHIMANO, B., AND MAYER, G. E., (1981). "Kinematic Control Equations for Simple Manipulators," in *IEEE Trans SMC-11*, pp. 449-55.

PRESS, W. H., FLANNERY, B. P., TEUKOLSKY, S. A., AND VETTERLING, W. T.(1986), *Numerical Recipes*, Cambridge University Press.

RAJAN, V. T., (1985). "Planning of Minimum-time Trajectories for Robot Arms," in *Proc. IEEE International conf. on Robotics and Automation*, pp. 759-64.

SAHAR, G. AND HOLLERBACH, J. M., (1986). "Planning of Minimum-time Trajectories for Robot Arms," in *Int. J. Robotics Research*, vol. 5, pp. 90-100.

SNYDER, W. E.(1985), , *Prentice-Hall.*

TAYLOR, R. H., (1979). "Planning and Execution of Straight Line Manipulator Trajectories," in *IBM J. of Res. & Dev.*, vol. 23, pp. 253-64.

THOMPSON, S. E. AND PATEL, R. V., (1987). "Formulation of Joint Trajectories for Industrial Robots Using B-Splines," in *IEEE Trans. IE-34*, pp. 192-99.

VANDERGRAFT, J. S.(1978), *Introduction to Numerical Computations*, Academic Press.

VUKOBRATOVIC, M. AND KIRCANSKI, M., (1982). "A Method for Optimal Synthesis of Manipulation Robot Trajectories," in *Trans. ASME, J. of Dyn. Syst., Meas. and Control*, vol. 104, pp. 188-93.

DE-BOOR, C.(1978), *A Practical Guide to Splines*, Springer-Verlag.

Appendix (A)
Acceleration Approach to the Formulation of Cubic Splines

Since $\theta(t)$ is assumed to be a cubic function, then its second derivative should be a linear function,

$$\ddot{\theta}(t) = \frac{\ddot{\theta}_i}{h_i}(t_{i+1}-t) + \frac{\ddot{\theta}_{i+1}}{h_i}(t-t_i) \qquad (A.1)$$

where,

$$h_i = t_{i+1} - t_i \qquad (A.2)$$

Now, integrating $(A.1)$ twice, and invoking the continuity conditions

$$\theta_{i-1}(t_{i-1}) = \theta_i(t_{i-1}) \qquad (A.3)$$

$$\theta_i(t_i) = \theta_{i+1}(t_i) \qquad (A.4)$$

yields,

$$\dot{\theta}(t) = \frac{\ddot{\theta}_i}{2h_i}(t_{i+1}-t)^2 + \frac{\ddot{\theta}_{i+1}}{2h_i}(t-t_i)^2 + \left[\frac{\theta_{i+1}}{h_i} - \frac{h_i \,.\, \ddot{\theta}_{i+1}}{6}\right] - \left[\frac{\theta_i}{h_i} - \frac{h_i \,.\, \ddot{\theta}_i}{6}\right] \qquad (A.5)$$

and,

$$\theta(t) = \frac{\ddot{\theta}_i}{6h_i}(t_{i+1}-t)^3 + \frac{\ddot{\theta}_{i+1}}{6h_i}(t-t_i)^3 + \left[\frac{\theta_{i+1}}{h_i} - \frac{h_i \,.\, \ddot{\theta}_{i+1}}{6}\right](t-t_i) - \left[\frac{\theta_i}{h_i} - \frac{h_i \,.\, \ddot{\theta}_i}{6}\right](t_{i+1}-t) \qquad (A.6)$$

By continuity of velocity,

$$\dot{\theta}_{i-1}(t_i) = \dot{\theta}_i(t_i) \qquad (A.7)$$

then,

$$h_{i-1}\ddot{\theta}_{i-1} + 2(h_i+h_{i-1})\ddot{\theta}_i + h_i\ddot{\theta}_{i+1} = 6\left[\frac{\theta_{i+1}-\theta_i}{h_i} - \frac{\theta_i-\theta_{i-1}}{h_{i-1}}\right] \qquad (A.8)$$

$$, \; i = 2,3,\ldots,m-1$$

where m is the number of points.

Thus $(A.8)$ gives a system of $n-2$ linear equations, which can be solved to obtain values for each $\ddot{\theta}_i$.

If the first and second derivatives at the boundaries are to be preset, the following is considered :

$$\ddot{\theta}_1 = \theta_{start} \tag{A.9}$$

$$\theta_2 = \theta_1 + h_1\dot{\theta}_1 + \frac{h_1^2}{3}\ddot{\theta}_1 + \frac{h_1^2}{6}\ddot{\theta}_2 \tag{A.10}$$

$$\theta_{n-1} = \theta_n - h_{n-1}\dot{\theta}_n + \frac{h_{n-1}^2}{3}\ddot{\theta}_n + \frac{h_{n-1}^2}{6}\ddot{\theta}_{n-1} \tag{A.11}$$

$$\ddot{\theta}_n = \theta_{end} \tag{A.12}$$

Substituting (*A*.9–12) into (*A*.8) enables the presetting of $\dot{\theta}_1$, $\dot{\theta}_n$, $\ddot{\theta}_1$ and $\ddot{\theta}_n$ by the user. Once the system is solved, and $\ddot{\theta}_3$ and $\ddot{\theta}_{n-2}$ are known, the values of θ_2 and θ_{n-1} can be calculated.

Appendix (B)

Quadratic Splines

Since $\theta(t)$ is a quadratic, then $\dot{\theta}(t)$ must be a linear function

$$\dot{\theta}(t) = \frac{\dot{\theta}_i}{h_i}(t_{i+1}-t) + \frac{\dot{\theta}_{i+1}}{h_i}(t-t_i) \tag{B.1}$$

$$, i=1,2,\ldots,m$$

where,

$$h_i = t_{i+1} - t_i \tag{B.2}$$

and m is the number of points.

Integrating $(B.1)$ yields

$$\dot{\theta}(t) = -\frac{\dot{\theta}_i}{2h_i}(t_{i+1}-t)^2 + \frac{\dot{\theta}_{i+1}}{2h_i}(t-t_i)^2 + C_1 \tag{B.3}$$

Evaluating C_1 yields

$$\theta(t_i) = \theta_i = -\frac{\dot{\theta}_i h_i}{2} + C_1 \tag{B.4}$$

$$\Rightarrow \quad C_1 = \theta_i + \frac{\dot{\theta}_i h_i}{2} \tag{B.5}$$

also,

$$\theta(t_{i+1}) = \theta_{i+1} = \frac{\dot{\theta}_{i+1} h_i}{2} + C_1 \tag{B.6}$$

$$\Rightarrow \quad C_1 = \theta_{i+1} - \frac{\dot{\theta}_{i+1} h_i}{2} \tag{B.7}$$

Equating (B.5) and (B.7) gives

$$\theta_{i+1} = \theta_i + \frac{h_i}{2}(\dot{\theta}_i + \dot{\theta}_{i+1}) \tag{B.8}$$

Appendix (C)

Velocity Approach to the Formulation of Cubic Splines

For an interval with only two points known, the *Hermit Interpolating Formula* can be expressed as

$$\theta(t) = \dot{\theta}_{k-1}\left[\frac{(t_k-t)^2(t-t_{k-1})}{h_k^2}\right] - \dot{\theta}_k\left[\frac{(t-t_{k-1})^2(t_k-t)}{h_k^2}\right] +$$

$$\theta_{k-1}\left[\frac{(t_k-t)^2[2(t-t_{k-1})+h_k]}{h_k^3}\right] + \theta_k\left[\frac{(t-t_{k-1})^2[2(t_k-t)+h_k]}{h_k^3}\right] \tag{C.1}$$

where,

$$h_k = t_{k+1} - t_k \tag{C.2}$$

then,

$$\ddot{\theta}(t_k) = \frac{2}{h_k}(\dot{\theta}_{k-1}+2\dot{\theta}_k) - 6\left[\frac{\theta_k-\theta_{k-1}}{h_k^2}\right] \tag{C.3}$$

$$\ddot{\theta}(t_{k+1}) = \frac{2}{h_{k+1}}(\dot{\theta}_k+2\dot{\theta}_{k+1}) - 6\left[\frac{\theta_{k+1}-\theta_k}{h_{k+1}^2}\right] \tag{C.4}$$

Equating (C.3) and (C.4) yields

$$\frac{2}{h_k}(\dot{\theta}_{k-1}+2\dot{\theta}_k) - 6\left[\frac{\theta_k-\theta_{k-1}}{h_k^2}\right] = \frac{2}{h_{k+1}}(\dot{\theta}_k+2\dot{\theta}_{k+1}) - 6\left[\frac{\theta_{k+1}-\theta_k}{h_{k+1}^2}\right] \tag{C.5}$$

Setting equal values for all values of h's and simplifying gives

$$\frac{3}{h^2}\theta_{k-1} - \frac{6}{h^2}\theta_k + \frac{3}{h^2}\theta_{k+1} = \frac{1}{h}[-\dot{\theta}_{k-1}-\dot{\theta}_k+2\dot{\theta}_{k+1}] \tag{C.6}$$

$$,\quad k=2,3,\ldots,l-1$$

where l is the number of points.

Since the values of θ_1 and θ_l are known, then for k=2,

$$-\frac{6}{h^2}\theta_2 + \frac{3}{h^2}\theta_3 = \frac{1}{h}[-\dot{\theta}_1-\dot{\theta}_2+2\dot{\theta}_3] - \frac{3}{h^2}\theta_1 \tag{C.7}$$

also for k=l–1,

$$\frac{3}{h^2}\theta_{l-2} - \frac{6}{h^2}\theta_{l-1} = \frac{1}{h}[-\dot{\theta}_{l-2}-\dot{\theta}_{l-1}+2\dot{\theta}_l] - \frac{3}{h^2}\theta_l \tag{C.8}$$

Thus, a system of $l-2$ equations can be solved to obtain the required position values θ_k, $k=1,2,\ldots,l-1$.

Appendix (D)

The PUMA 600 Model Data

Table (D.1) : Link Coordinate Parameters

Joint i	ϕ_i *(degree)*	α_i *(degree)*	a_i *(mm)*	d_i *(mm)*	θ Range *(degree)*
1	90	-90	0	0	-160 to 160
2	0	0	431.8	149.09	-225 to 45
3	90	90	-20.32	0	-45 to 225
4	0	-90	0	433.07	-110 to 170
5	0	90	0	0	-100 to 100
6	0	0	0	56.25	-266 to 266

Table (D.2) : Masses and Centres of Masses

Joint#	Link Mass (kg)	Centre of Mass (cm)		
		x	y	z
1	2.27	0	0	7.3
2	15.91	-43.18	0	0
3	6.82	0	0	10
4	3.18	0	10	0
5	0.91	0	0	1.
6	0.45	0	0	3.
6 (loaded)	2.75	0	0	8.018

Table (D.3) : Radii of Gyration			
Joint#	Radii of Gyration (cm^2)		
	x	y	z
1	84.58	170.9	117.6
2	62.85	2323.0	2369.0
3	132.6	416.7	329.2
4	106.3	3.145	103.1
5	33.97	33.97	4.396
6	120.1	120.1	6.667
6 (loaded)	92.40	92.40	6.109

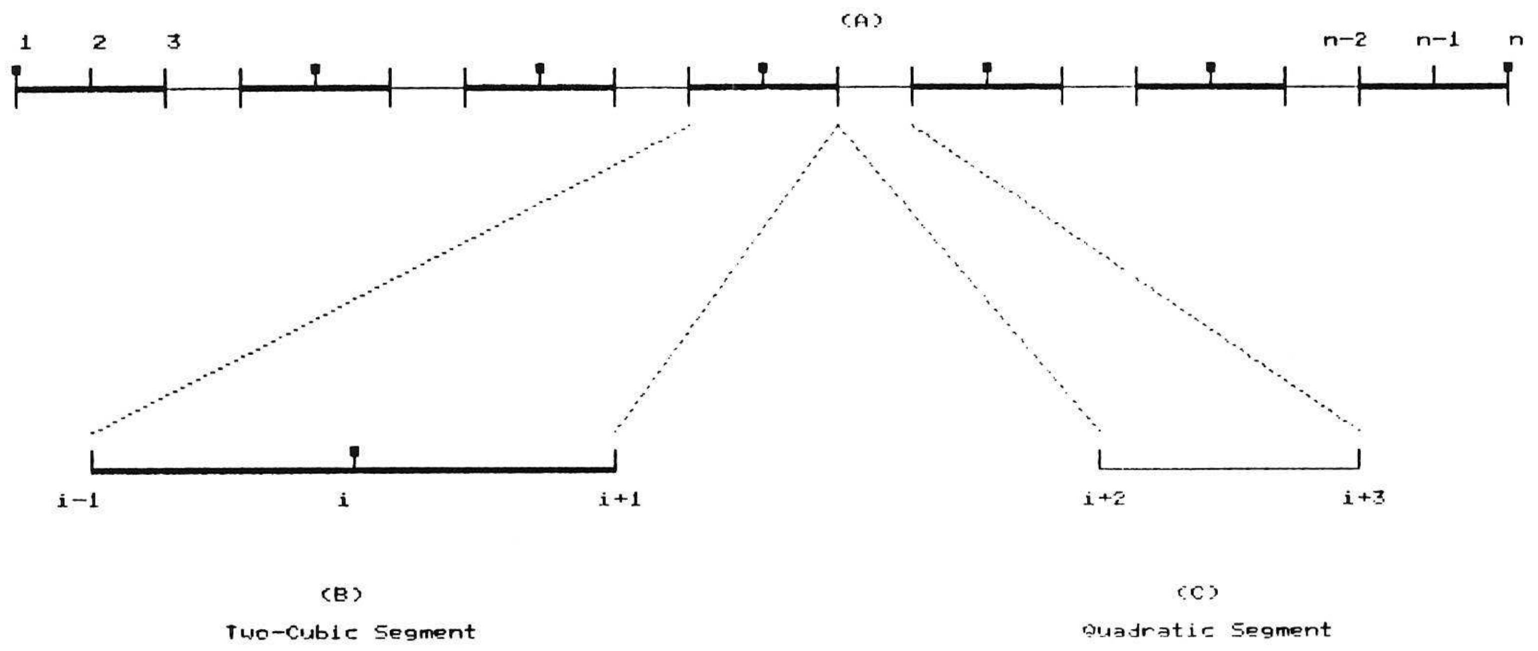


FIGURE (1)
The Constructed Trajectory

THE OCQS ALGORITHM			
Title The Constructed Trajectory			
Size A	Document Number 1		REV
Date: December 13, 1988	Sheet	of	

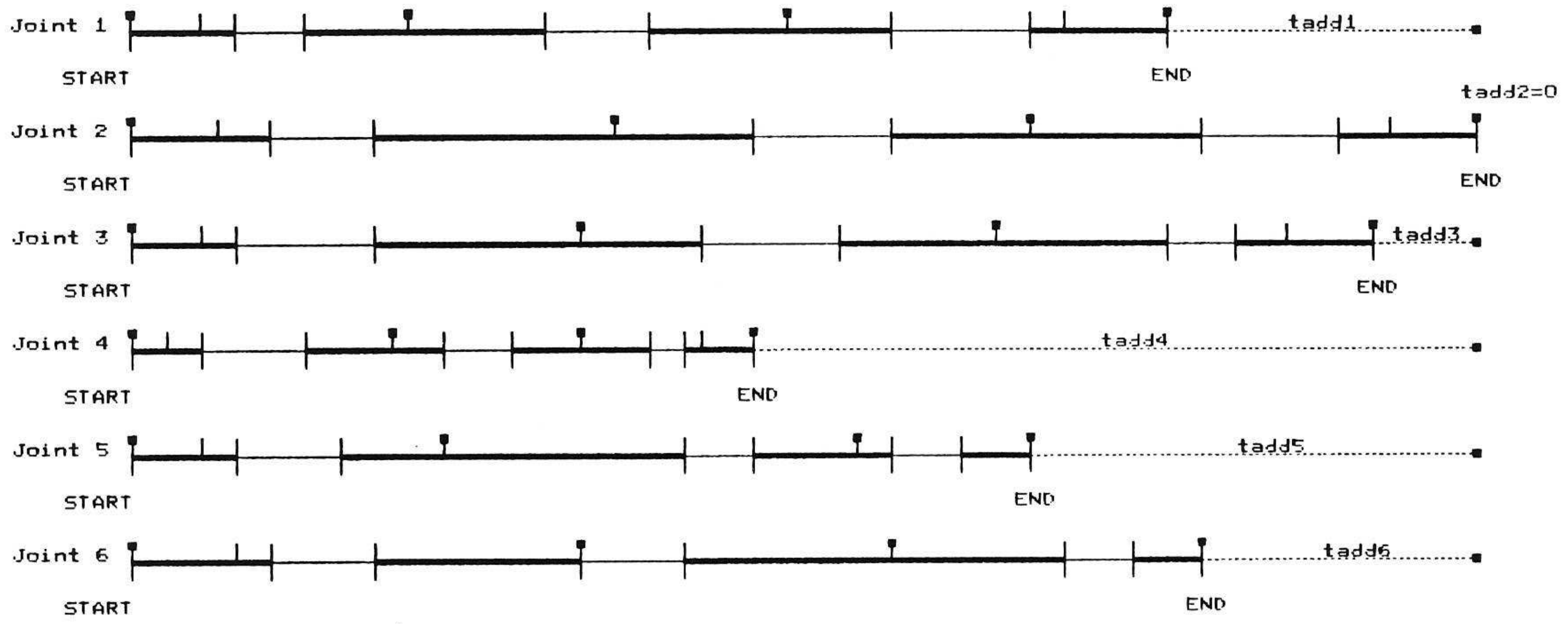


FIGURE (2)

THE OCQS ALGORITHM		
Title FIGURE (2)		
Size A	Document Number 2	REV
Date: November 28, 1988	Sheet	of

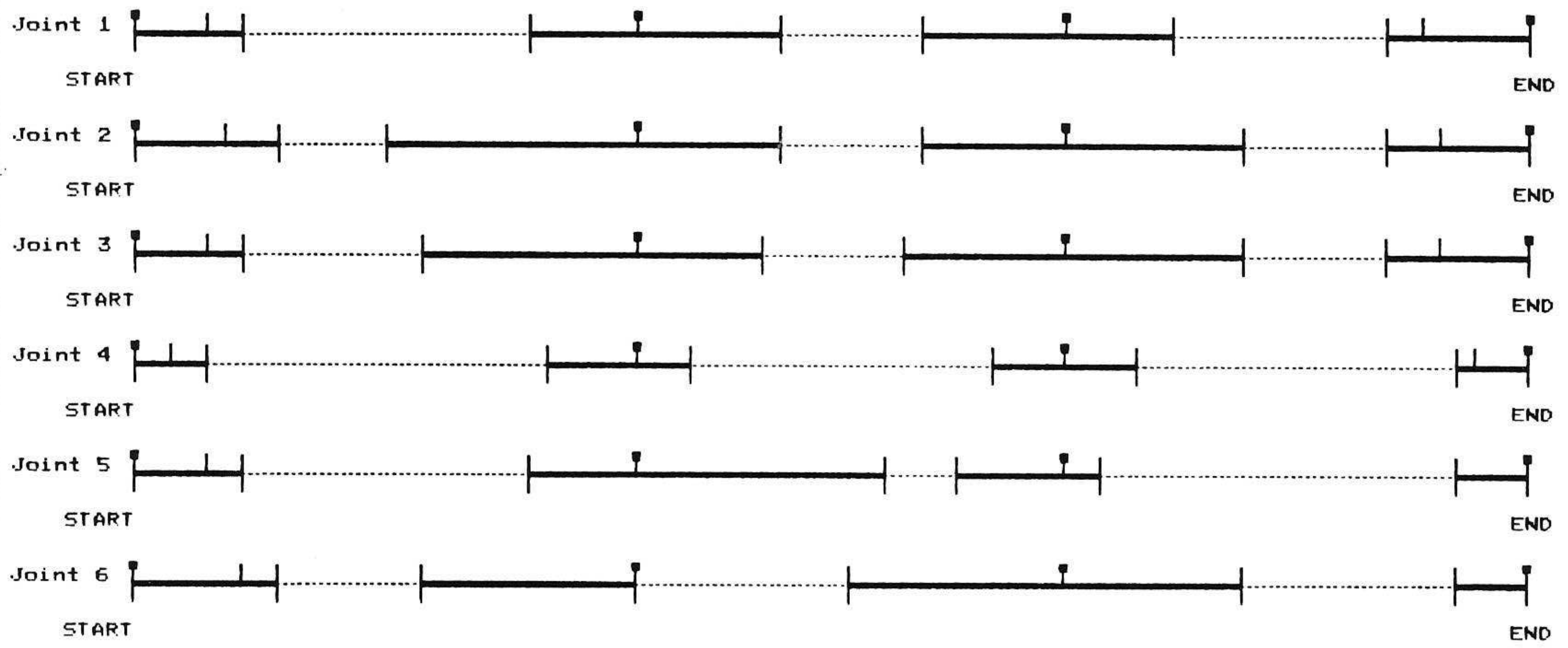


FIGURE (3)

THE OCQS ALGORITHM		
Title	FIGURE (3)	
Size A	Document Number 3	REV
Date: November 28, 1988	Sheet	of

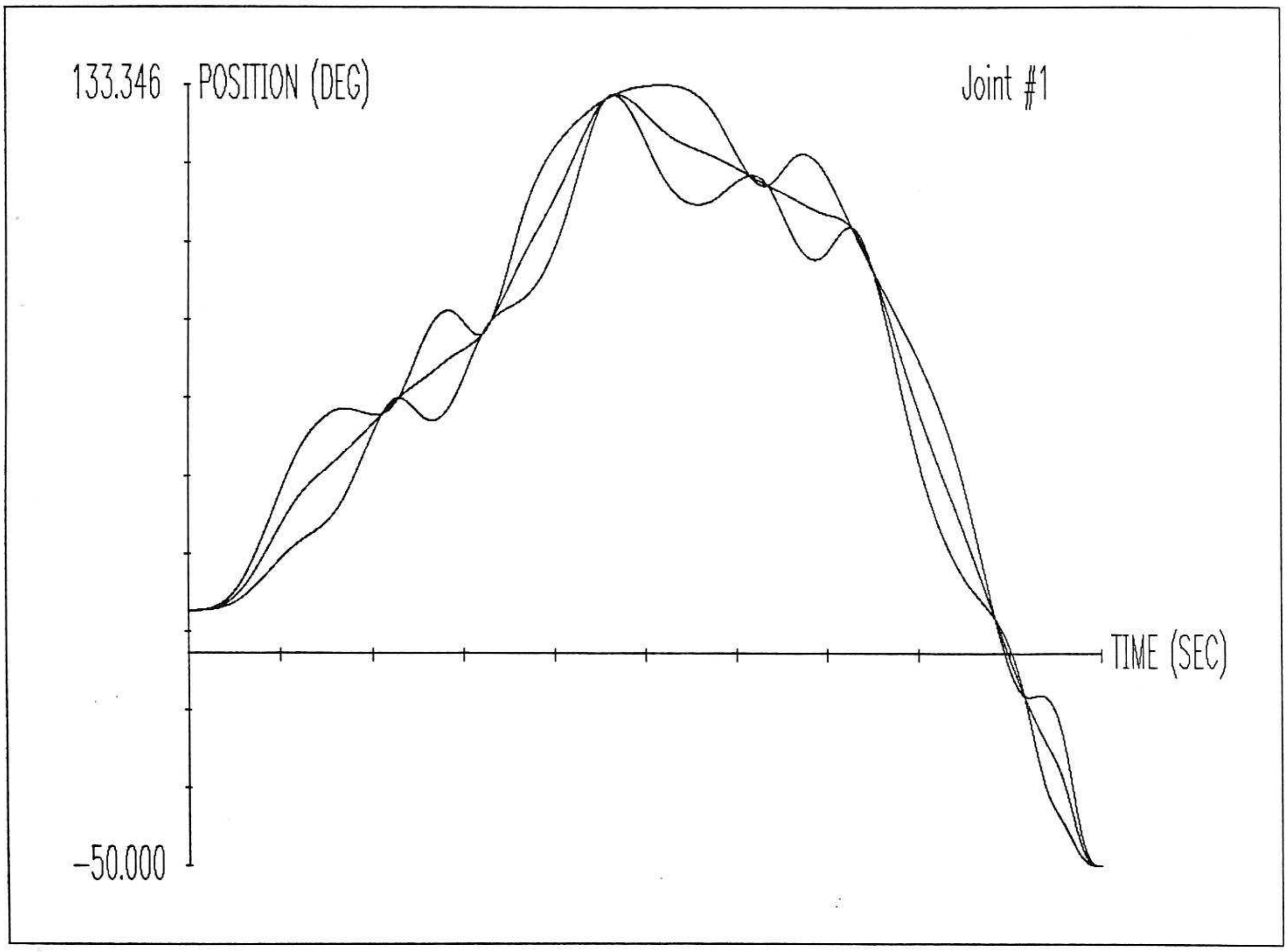


Figure 4

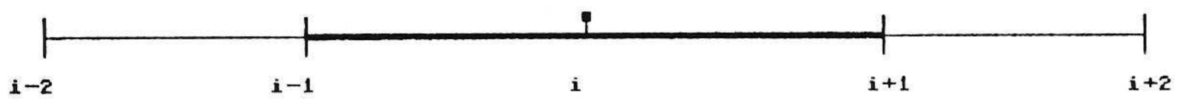


FIGURE (5)

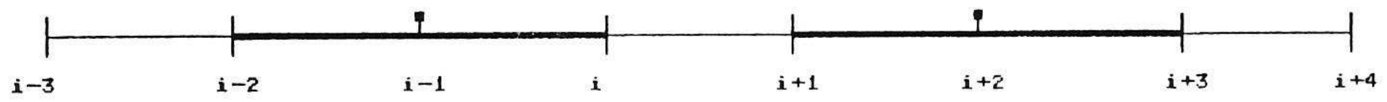


FIGURE (6)

THE OCQS ALGORITHM			
Title FIGURES (5),(6)			
Size	Document Number		REV
A	4		
Date: December 1, 1988		Sheet of	

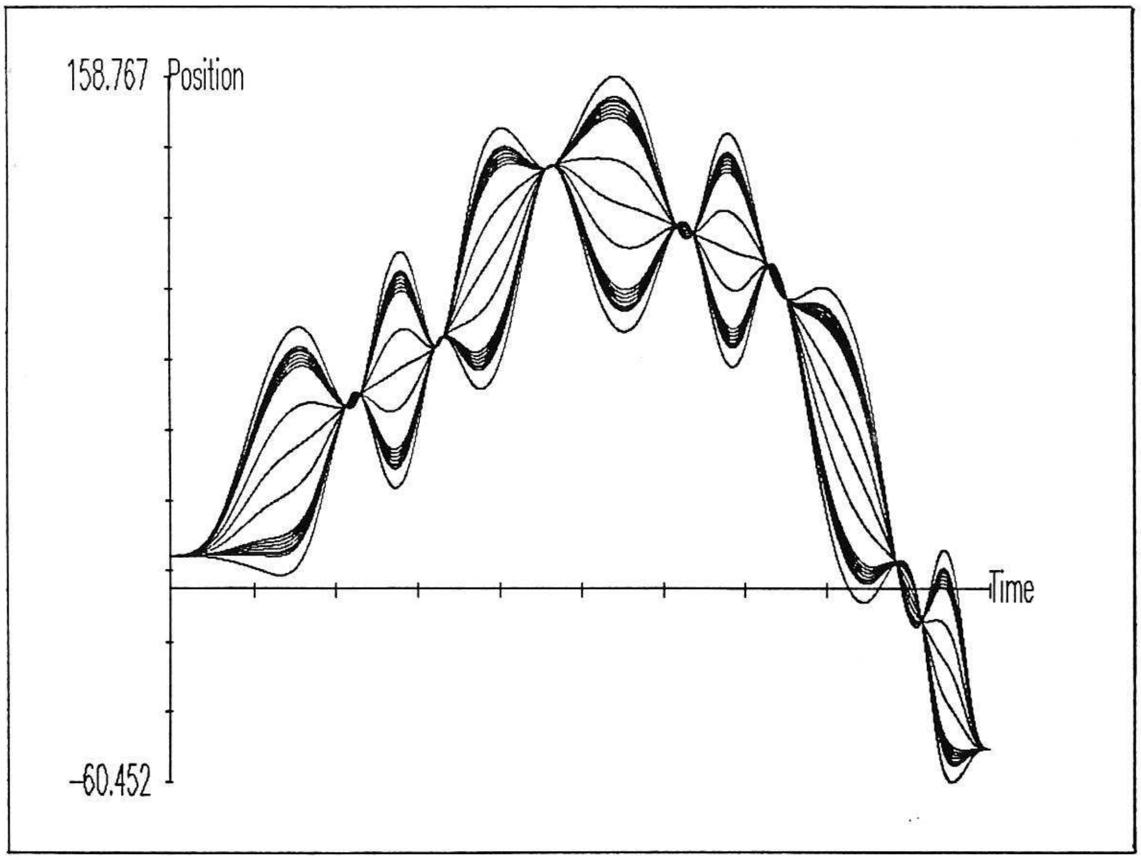


Figure 7

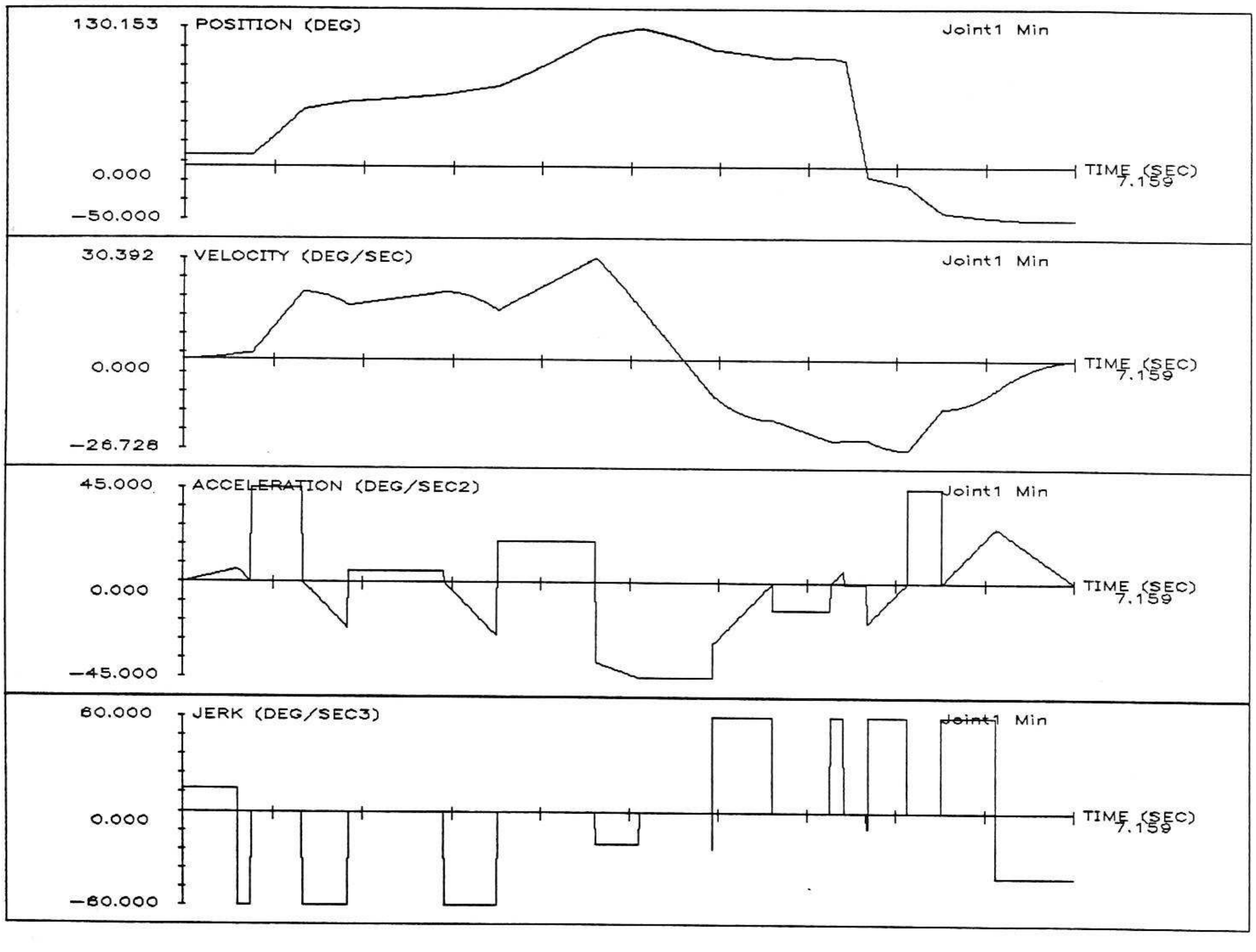
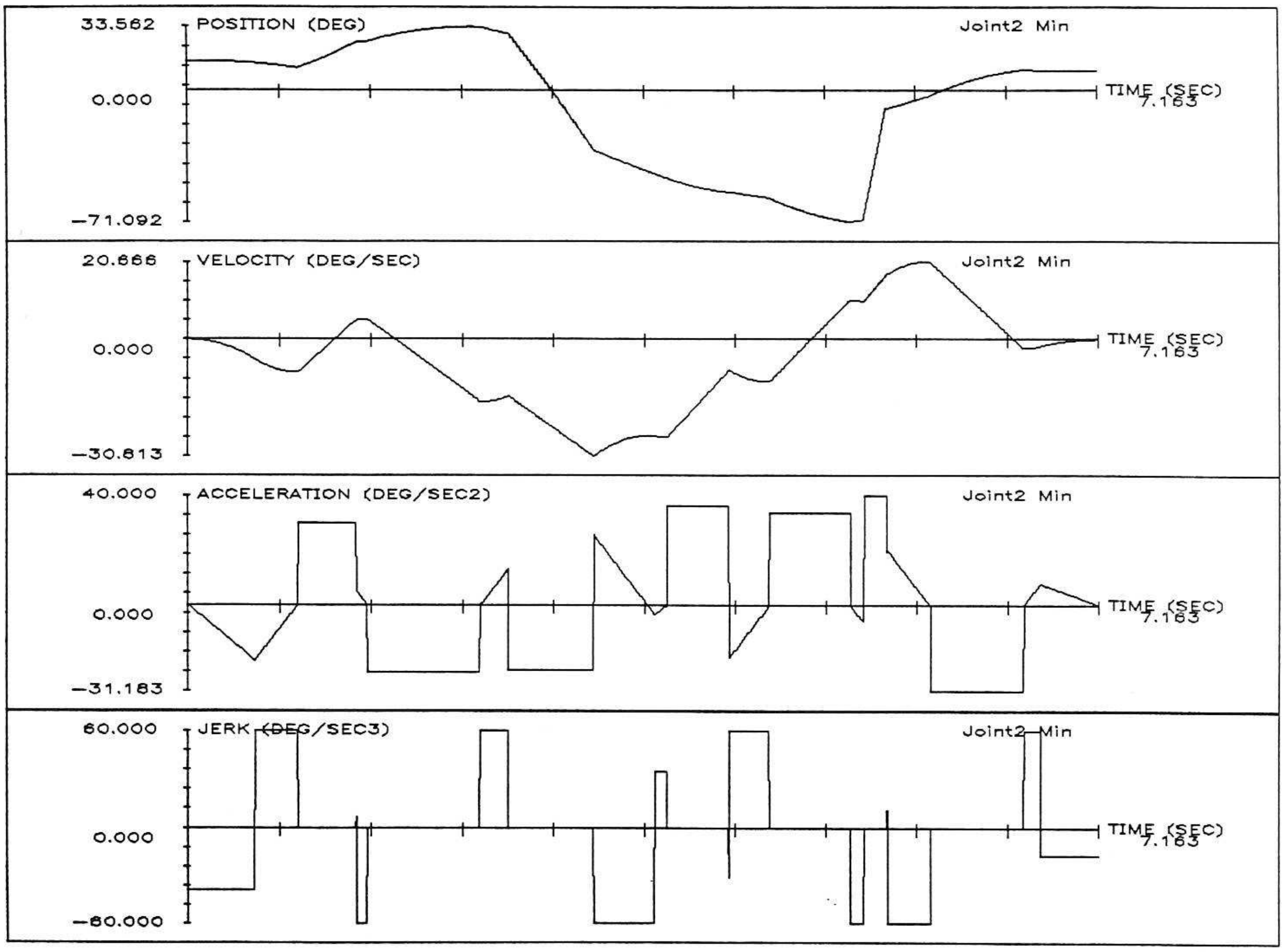
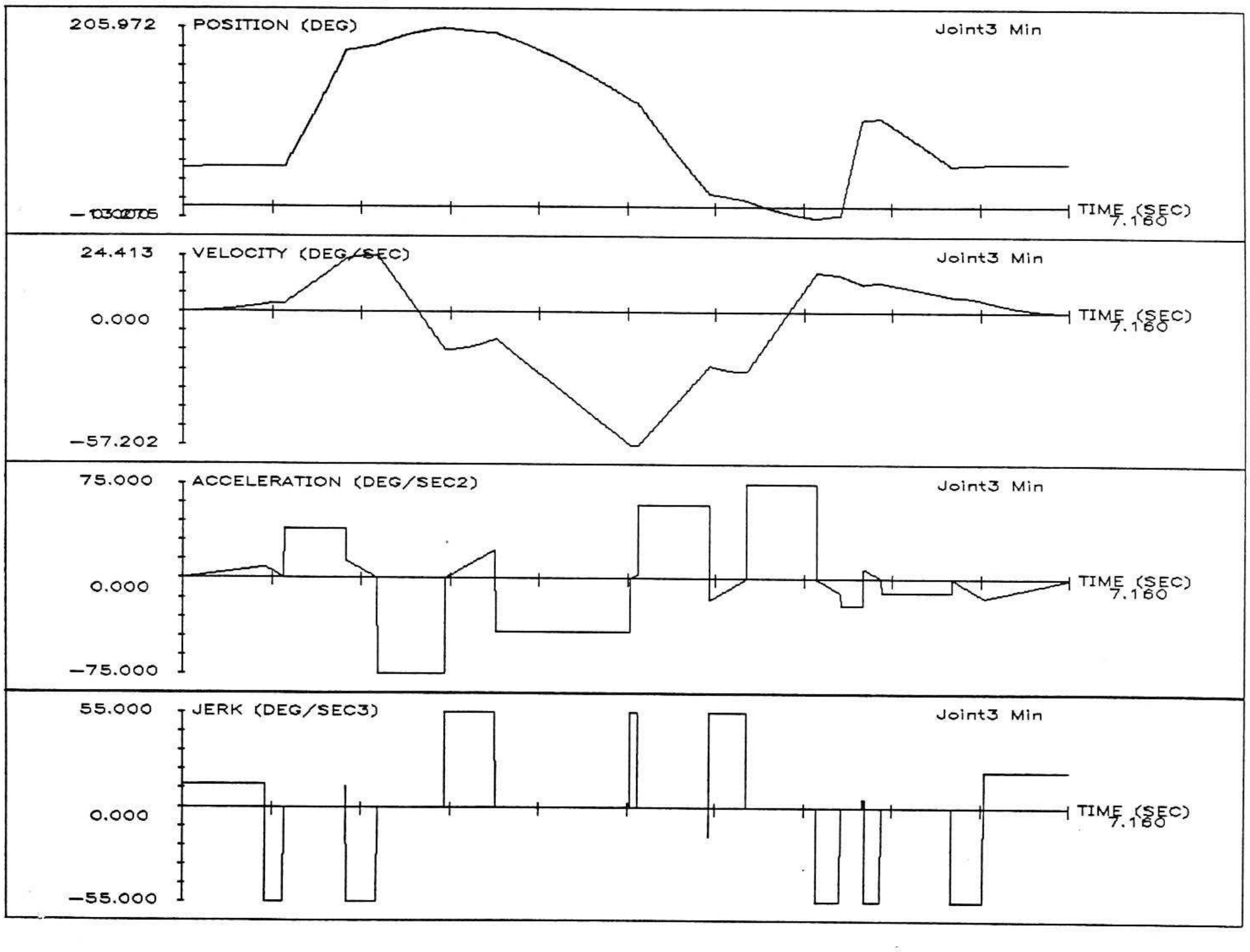

130.153
POSITION (DEG)
Joint1 Min
0.000
TIME (SEC)
7.159
-50.000
30.392
VELOCITY (DEG/SEC)
Joint1 Min
0.000
TIME (SEC)
7.159
-26.728
45.000
ACCELERATION (DEG/SEC2)
Joint1 Min
0.000
TIME (SEC)
7.159
-45.000
60.000
JERK (DEG/SEC3)
Joint1 Min
0.000
TIME (SEC)
7.159
-60.000

Figure 8



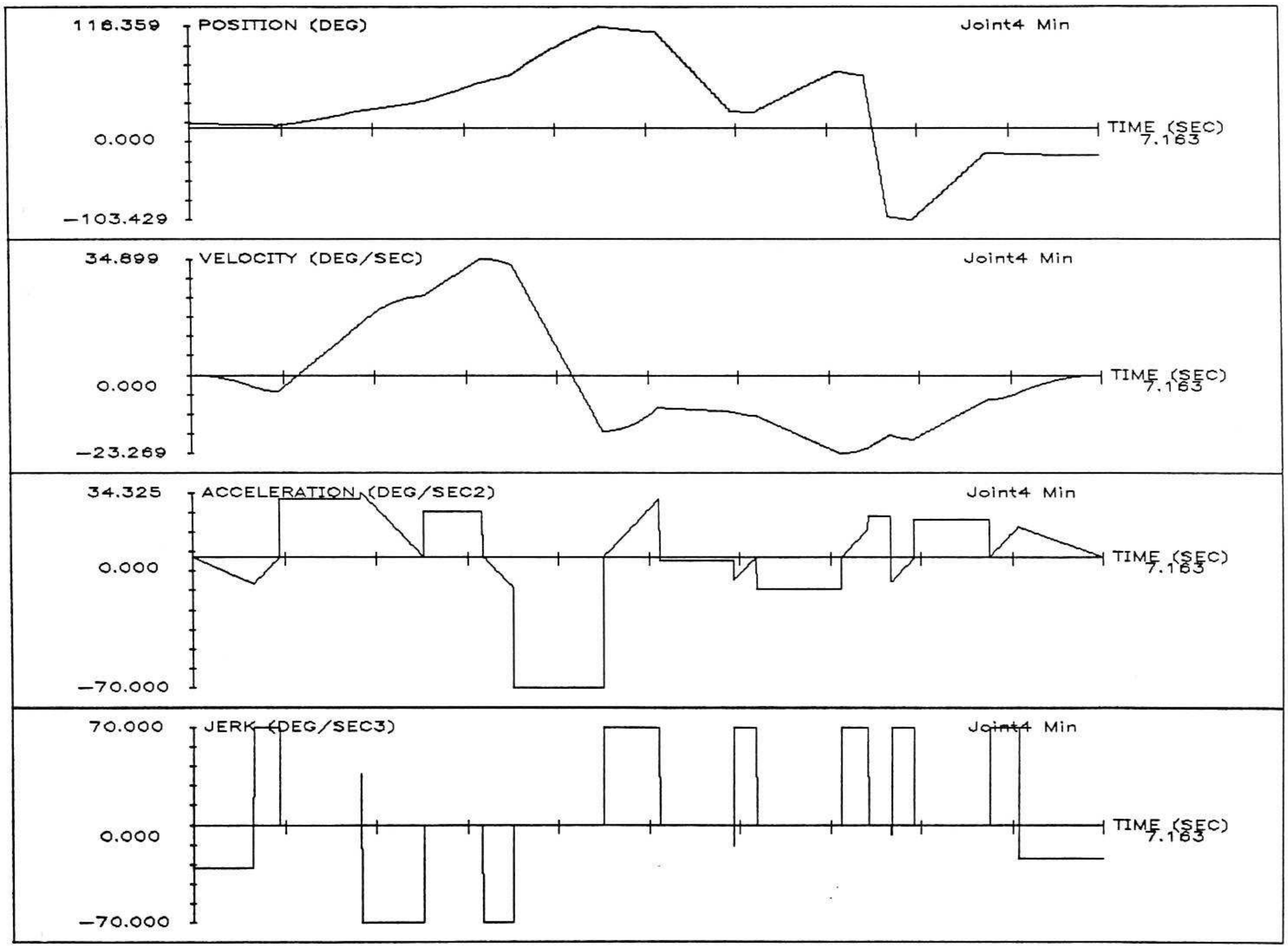
33.562
POSITION (DEG)
Joint2 Min
0.000
TIME (SEC)
7.163
-71.092
20.666
VELOCITY (DEG/SEC)
Joint2 Min
0.000
TIME (SEC)
7.163
-30.813
40.000
ACCELERATION (DEG/SEC2)
Joint2 Min
0.000
TIME (SEC)
7.163
-31.183
60.000
JERK (DEG/SEC3)
Joint2 Min
0.000
TIME (SEC)
7.163
-60.000

Figure 9



205.972
POSITION (DEG)
Joint3 Min
TIME (SEC)
7.160
24.413
VELOCITY (DEG/SEC)
Joint3 Min
0.000
TIME (SEC)
7.160
-57.202
75.000
ACCELERATION (DEG/SEC2)
Joint3 Min
0.000
TIME (SEC)
7.160
-75.000
55.000
JERK (DEG/SEC3)
Joint3 Min
0.000
TIME (SEC)
7.160
-55.000

Figure 10



116.359
POSITION (DEG)
Joint4 Min
0.000
TIME (SEC)
7.163
-103.429
34.899
VELOCITY (DEG/SEC)
Joint4 Min
0.000
TIME (SEC)
7.163
-23.269
34.325
ACCELERATION (DEG/SEC2)
Joint4 Min
0.000
TIME (SEC)
7.163
-70.000
70.000
JERK (DEG/SEC3)
Joint4 Min
0.000
TIME (SEC)
7.163
-70.000

Figure 11

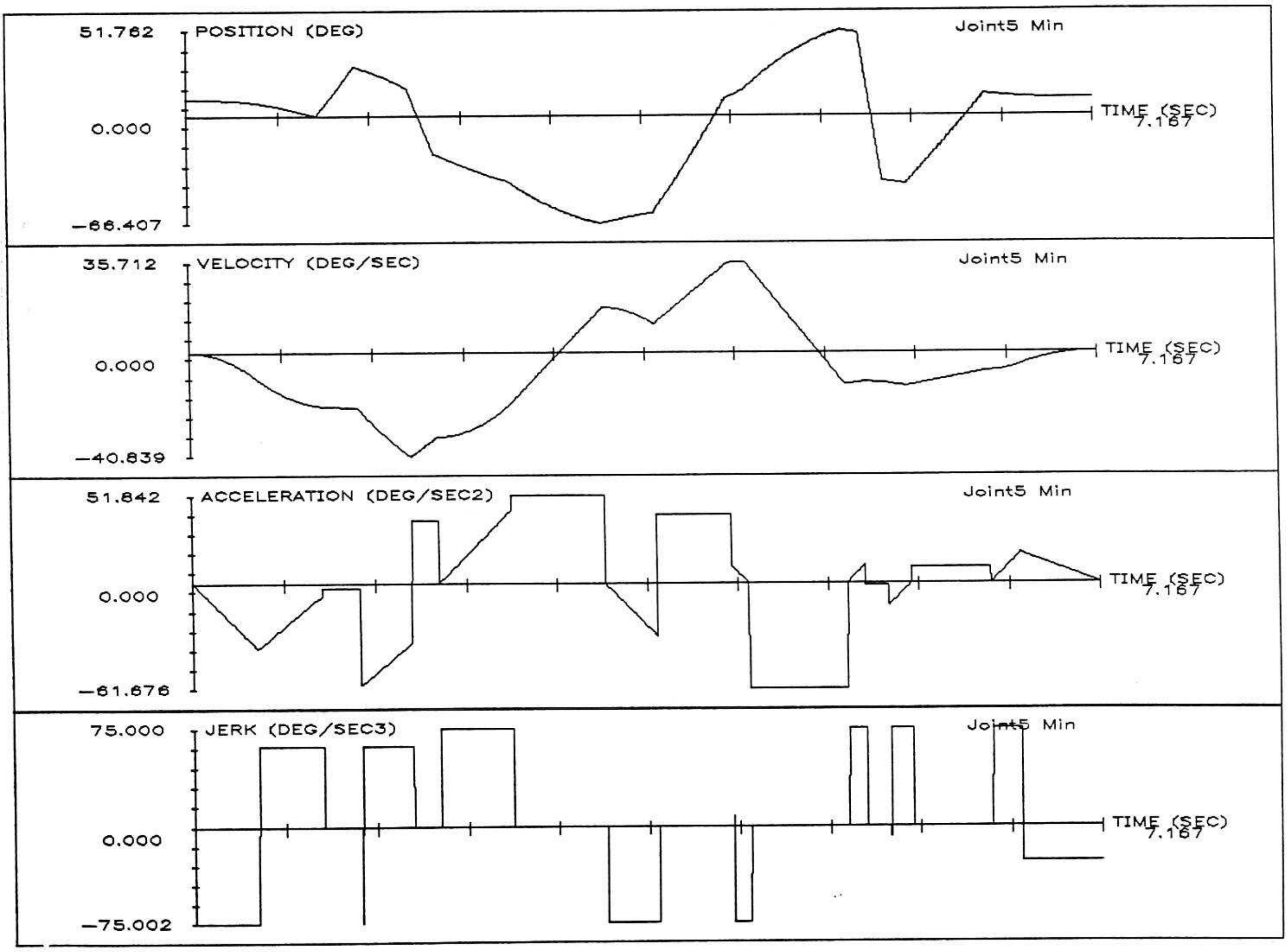
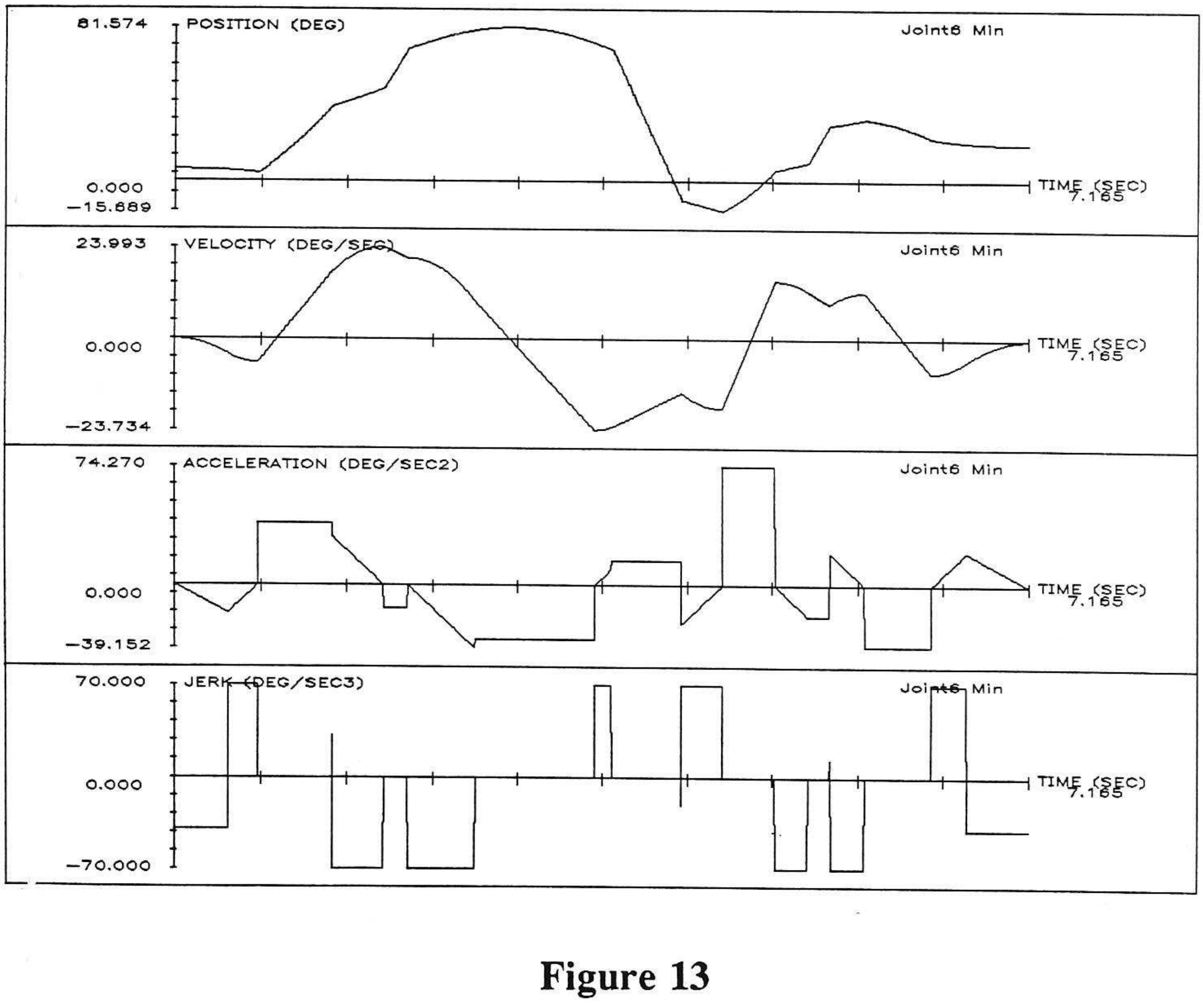

51.762
POSITION (DEG)
Joint5 Min
0.000
TIME (SEC)
7.167
-66.407
35.712
VELOCITY (DEG/SEC)
Joint5 Min
0.000
TIME (SEC)
7.167
-40.839
51.842
ACCELERATION (DEG/SEC2)
Joint5 Min
0.000
TIME (SEC)
7.167
-61.676
75.000
JERK (DEG/SEC3)
Joint5 Min
0.000
TIME (SEC)
7.167
-75.002

Figure 12



81.574
POSITION (DEG)
Joint6 Min
0.000
-15.889
TIME (SEC)
7.165
23.993
VELOCITY (DEG/SEC)
Joint6 Min
0.000
-23.734
TIME (SEC)
7.165
74.270
ACCELERATION (DEG/SEC2)
Joint6 Min
0.000
-39.152
TIME (SEC)
7.165
70.000
JERK (DEG/SEC3)
Joint6 Min
0.000
-70.000
TIME (SEC)
7.165

Figure 13

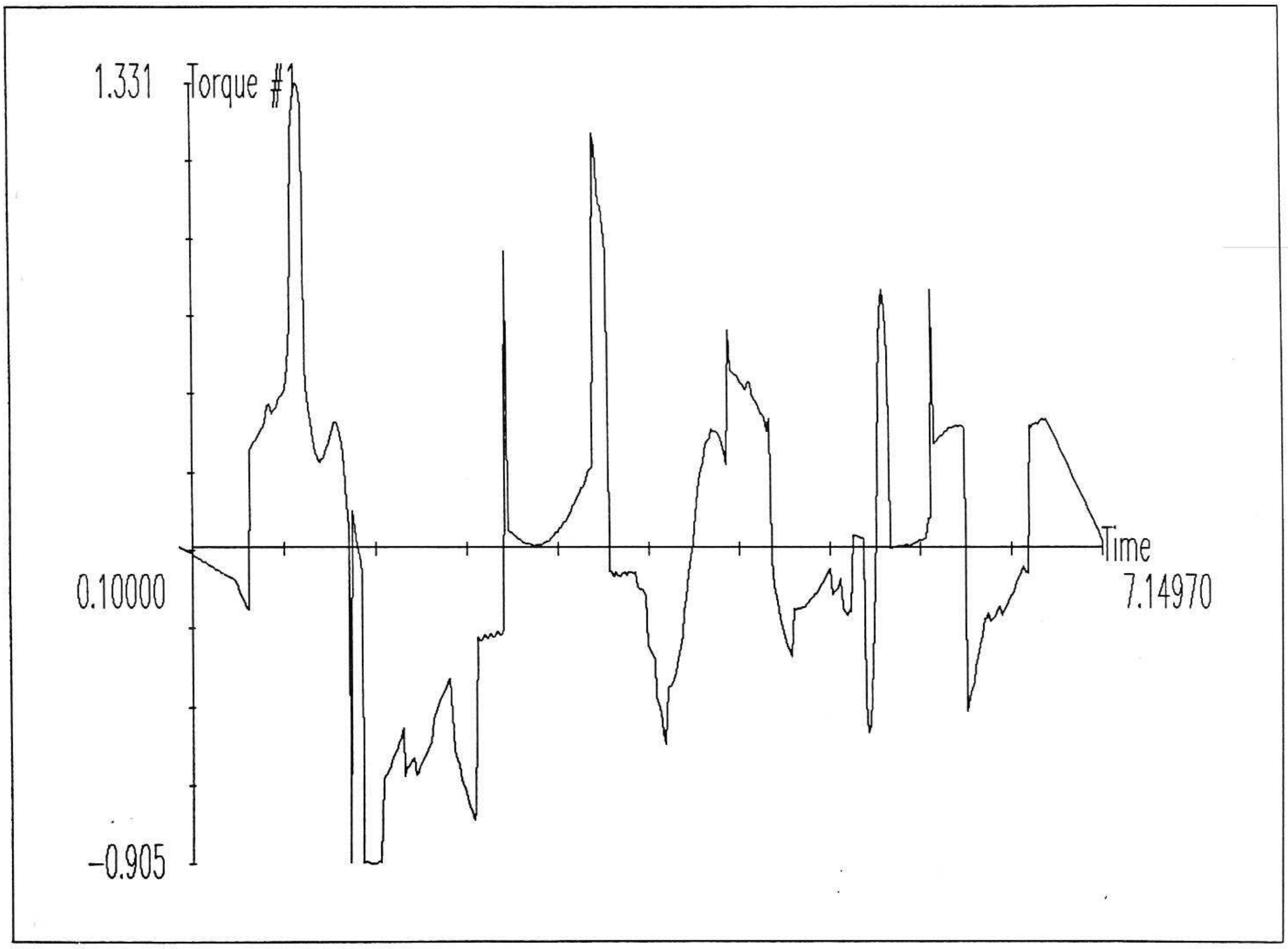


Figure 14