

# The Generation and Optimization of Trigonometric Joint Trajectories for Robotic Manipulators

D. Simon and C. Isik

Department of Electrical Engineering  
Syracuse University, Syracuse, NY 13244-1240

## Abstract

Interpolation of a sequence of desired robot configurations is realized using trigonometric splines. This original application has several advantages over existing methods (e.g. those using algebraic splines). For example, the computational expense is lower, more constraints can be imposed on the trajectory, and smoother trajectories are obtained using the proposed approach. The paper introduces a trajectory interpolation algorithm, discusses methods for both closed-form and numerical path optimization, and includes examples.

## 1. Introduction

This paper presents an original application of trigonometric interpolation to robotic manipulator path planning. The specific type of trigonometric interpolation used is piecewise trigonometric interpolation, also referred to in this paper as trigonometric splines.

The position and orientation of the end effector of a robotic manipulator can be computed analytically from the joint angles using manipulator kinematics [1]. The joint angles of a robotic manipulator can be computed from the Cartesian position and orientation of its end effector using inverse kinematics. Inverse kinematics solutions cannot be computed analytically, but must be computed numerically.

In specifying a desired motion of a manipulator, it is common to specify the motion of the end effector as a curve in 3-dimensional Cartesian space. Then inverse kinematics is used at specific points along the curve to generate a sequence of joint angles. A greater number of joint angles results in a more accurate trajectory. But it also results in a larger computational effort due to the increased number of inverse kinematics solutions required.

The discrete sequence of joint angles is then input to some interpolation algorithm to generate joint angles as continuous functions of time. The interpolation has typically been accomplished using algebraic cubic splines [7] or algebraic quartic splines [10]. The use of cubic splines ensures continuity of the first 2

derivatives. The use of quartic splines ensures continuity of the first 3 derivatives, but also results in greater oscillations between the knots. Continuity of derivatives is desirable to reduce the wear and tear on the manipulator. Continuity of even higher order derivatives can be ensured by using higher order algebraic splines, but this is at the expense of large oscillations near the endpoints of the trajectory [3].

Once the joint angles have been determined as continuous functions of time, joint angles are input to the feedback control system of the robot at the controller rate.

Section 2 of this paper gives a brief history and overview of trigonometric splines. Section 3 discusses how trigonometric splines can be used to plan a manipulator path, and Section 4 gives the trajectory formulation algorithm. It is shown that the use of trigonometric functions results in a large savings of computational effort compared to other path planning methods (e.g. compared to the use of algebraic splines). It is also shown that the use of trigonometric splines allows the path planning algorithm to proceed one segment at a time, so the path can be altered in mid-course if necessary (e.g. if an obstacle needs to be avoided). Section 5 discusses methods of optimizing the trigonometric path, and Section 6 presents some numerical examples and comparisons with algebraic splines. It is seen that the use of trigonometric splines not only allows more constraints on the trajectory, but also results in lower velocities, accelerations, and jerks. Section 7 presents some concluding remarks.

## 2. Trigonometric Splines

The term *trigonometric spline* was first introduced by Schoenberg [9]. Schoenberg defined an  $m^{\text{th}}$ -order trigonometric spline function as a function  $y(t)$  with the following properties.

1.  $y(t)$  is periodic with period  $2\pi$ , and is  $4m$  times continuously differentiable.
2.  $y(t)$  satisfies the interpolation conditions  $y(t_j) = y_j$  ( $j = 0, \dots, n$ ), where  $t_j \in [0, 2\pi)$ .

3. In each of the  $n$  closed arcs  $[t_j, t_{j+1}]$  the function  $y(t)$  is an element of  $\text{Span}[1, \cos(rt), \sin(rt), t \cos(rt), t \sin(rt)]$ , ( $r = 1, \dots, m$ ).

The last property listed above shows that trigonometric splines are *not* composed solely of trigonometric functions. Schoenberg showed that given the real numbers  $(t_0, \dots, t_n) \in [0, 2\pi)$  and  $(y_0, \dots, y_n)$ , there is a unique  $m^{\text{th}}$ -order trigonometric spline function which satisfies  $y(t_j) = y_j$ , ( $j = 0, \dots, n$ ). Furthermore, he showed that of *all* functions  $f(t)$  which have period  $2\pi$ , which are  $4m$  times continuously differentiable, and which satisfy  $f(t_j) = y_j$ , the trigonometric spline function satisfies the following minimal property.

$$\int_0^{2\pi} (\Delta_m f(t))^2 dt = \text{minimum} \quad (1)$$

where  $\Delta_m$  is the differential operator

$$\Delta_m = D(D^2 + 1^2)(D^2 + 2^2) \dots (D^2 + m^2). \quad (2)$$

Since Schoenberg's paper, other papers [4, 8] have defined an  $m^{\text{th}}$ -order trigonometric spline function as a function  $y(t)$  with the following properties.

1.  $y(t)$  is periodic.
2. If  $y(t)$  has a total of  $2m$  constraints in each of the  $n$  closed arcs  $[t_{i-1}, t_i]$  ( $i = 1, \dots, n$ ), the function  $y(t)$  has the form

$$y(t) = a_{i,0} + \sum_{k=1}^{m-1} (a_{i,k} \cos kt + b_{i,k} \sin kt) + a_{i,m} \cos m(t - \gamma_i) \quad (3)$$

for  $t_{i-1} < t < t_i$ , where

$$\gamma_i = \sum_{j=0}^{2m-1} \tau_{i,j} / 2m, \quad (4)$$

and  $\tau_{i,j}$  are the values of  $t$  where the  $i^{\text{th}}$  spline segment has a constraint applied. This form for  $y(t)$  guarantees the uniqueness and existence of a solution [4, 5].

So the term *trigonometric spline* is not a well-defined term. Because of the inconsistent nomenclature in the literature, this paper will use the term *Schoenberg splines* to refer to Schoenberg's definition, and *trigonometric splines* to refer to the more natural definition.

The above formula for  $y(t)$  shows that there are exactly  $2m$  coefficients for *each segment* of the trigonometric spline. Thus it is reasonable to impose  $2m$  constraints on each segment. We will choose the constraints to be  $y^{(r)}(t_i) = y_i^{(r)}$ , ( $r = 0, \dots, m-1$ ),

( $i = 0, \dots, n$ ). Of course, we require  $y^{(r)}(t_i^-) = y^{(r)}(t_i^+)$  if the trigonometric spline and its first  $m-1$  derivatives are to be continuous. But by constraining the actual values of  $y^{(r)}(t_i)$  rather than simply requiring continuity of  $y^{(r)}(t)$ , the determination of the coefficients is decoupled for each trigonometric polynomial. This results in a large savings of computational effort.

The formula for  $y(t)$  shows that trigonometric splines are a type of hermite interpolation. (A hermite interpolant is one which interpolates both function values and derivative values, while a lagrange interpolant is one which interpolates only function values.) There are well known relationships between the lagrange trigonometric polynomial coefficients which interpolate a given function, and the Fourier Series coefficients which approximate that function [3]. But there is no known analogus relationship between hermite trigonometric polynomial coefficients and Fourier Series coefficients.

### 3. Trigonometric Splines Applied to Path Planning

A desired continuous time Cartesian trajectory can be discretized into  $(n+1)$  Cartesian goal points at times  $0 = t_0 < t_1 < \dots < t_n$ . Then inverse kinematics can be performed at each of these goal points, resulting in the set of  $(n+1)$  joint space goal points  $\theta(t_i)$ .

Then  $n$   $4^{\text{th}}$ -order trigonometric polynomials  $y_i(t)$  can be generated. The function  $y_i(t)$  is defined *only* on the time interval  $[t_{i-1}, t_i]$ . These  $n$  trigonometric polynomials are joined together to form a trigonometric spline. Since  $y_i(t)$  is a  $4^{\text{th}}$ -order trigonometric polynomial, it has 8 undetermined coefficients. The 8 constraints used to determine the coefficients of  $y_i(t)$  are the values of the function and its first three derivatives at the two endpoints  $t_{i-1}$  and  $t_i$ . The function values at the endpoints are given by the inverse kinematics solution of the Cartesian trajectory at those two points. There are several different ways to specify the first three derivatives at the two endpoints. One way is that the user may desire certain joint derivatives at the knots. Another possibility is that these constraints could be chosen to minimize some objective function (see Section 5).

The 8 constraints and 8 unknown coefficients for spline segment  $y_i(t)$  require the inversion of an  $8 \times 8$  matrix ( $A_i$ ) in order to determine the coefficients for  $y_i(t)$ . But this matrix inversion can be performed a priori. It does not need to be performed real-time. The  $8 \times 8$  matrix  $A_i$  is a function of only 2 parameters:  $t_{i-1}$  and  $t_i$ . So the time interval of *each* spline segment can be normalized to a *fixed*  $t_{i-1}$  and  $t_i$ .

This is the key to the computational benefit of using trigonometric splines. In contrast to algebraic

splines, there is *no* need to solve a set of linear equations in order to determine the spline coefficients. When using trigonometric splines, the spline coefficients can be solved by simply multiplying a known matrix ( $A_i^{-1}$ , which is independent of  $i$ ) by a vector composed of knot angles and derivatives.

It can be shown that  $t_{i-1} = 0$  and  $t_i = \pi/4$ , ( $i = 1, \dots, n$ ), give the smoothest (in general) trajectories (under the constraint that matrix  $A_i$  is well-conditioned). Then Equation 4 shows that  $\gamma_i = \pi/8$  for all  $i$ , and Equation 3 becomes the familiar system

$$y_i(t) = a_{i,0} + \sum_{k=1}^3 (a_{i,k} \cos kt + b_{i,k} \sin kt) + a_{i,m} \cos mt \quad (5)$$

for  $0 < t < \pi/4$ , ( $i = 1, 2, \dots, n$ ).

#### 4. The Trajectory Formulation Algorithm

The input to the algorithm is a set of  $(n+1)$  joint space goal points  $\theta_i$  ( $i = 0, \dots, n$ ) which have been determined by an inverse kinematics algorithm. The normalized time interval for each spline segment is chosen to be  $[0, \pi/4]$ .

1. Determine the  $3(n+1)$  constraints

$$y_i^{(r)}(t_i) = y_i^{(r)} \quad (i = 0, \dots, n) \quad (r = 1, 2, 3). \quad (6)$$

Note that while each normalized spline segment  $y_i(t)$  has length  $\pi/4$ , the desired path length is  $T$ . So the derivatives of the scaled trajectory are related to the derivatives of the derivatives of the actual trajectory as follows.

$$\theta_i^{(r)}(t) = (n\pi/4T)^r y_i^{(r)}(n\pi t/4T). \quad (7)$$

2. Use the constraints for each normalized spline segment  $y_i(t)$  to determine the 8 coefficients of  $y_i(t)$  in Equation 5.

$$\begin{pmatrix} a_{i,0} \\ a_{i,1} \\ b_{i,1} \\ a_{i,2} \\ b_{i,2} \\ a_{i,3} \\ b_{i,3} \\ a_{i,4} \end{pmatrix} = A_i^{-1} \begin{pmatrix} \theta_{i-1} \\ \theta_i \\ y'_{i-1} \\ y'_i \\ y''_{i-1} \\ y''_i \\ y'''_{i-1} \\ y'''_i \end{pmatrix} \quad (i = 1, 2, \dots, n). \quad (8)$$

3. Define the time-scaled trigonometric spline as

$$y(t + (i-1)\pi/4) = y_i(t), \quad (9)$$

$$t \in [0, \pi/4], \quad (i = 1, \dots, n).$$

The function  $y(t)$  is a trigonometric spline which satisfies the desired interpolation conditions, and which has length  $n\pi/4$  seconds.

4. Unscale the spline so that the path length is converted from its scaled length  $n\pi/4$  to its desired length  $T$ .

$$\theta(t) = y(n\pi t/4T). \quad (10)$$

5. Now  $\theta(t)$  is a continuous time joint angle function. Steps 1-4 above need to be performed once for each robot joint.

Note that if one knot angle needs to be changed (e.g. due to an obstacle) the spline segments which do not touch that knot do not need to be changed. If an interior knot angle changes, only 2 spline segments need to be changed. The other  $n-2$  spline segments can remain as they are. This makes real-time obstacle avoidance a computationally inexpensive procedure.

#### 5. Optimization

The user of the trajectory formulation algorithm described in the previous section is free to choose the first three trajectory derivatives at each knot. The user will typically desire to set the derivatives at the endpoints to zero. A simple and reasonable heuristic method of choosing the derivatives at the interior knots would be to use some central difference methods on the knot angles.

However, if additional computer time is available, the knot derivatives can be chosen to minimize some objective function. A general objective function can be written in the form

$$J = f(\vec{\theta}(t)) \quad (11)$$

where  $f(\cdot)$  is a general nonlinear function,  $\vec{\theta}(t)$  is a  $P$ -vector of trigonometric splines, and  $P$  is the number of joints that the robot has.

Recall that  $\vec{\theta}(t)$  is a time-scaled version of  $\vec{y}(t)$  (see Equation 10) where  $\vec{y}(t)$  is the  $P$ -vector of normalized trigonometric splines, and  $\vec{y}(t)$  is composed of the  $n$  spline segments  $\vec{y}_i(t)$  (see Equation 9). Therefore Equation 11 can be written as

$$J = g\left[\sum_{k=1}^n \vec{y}_k(t)\right] \quad (12)$$

where  $g(\cdot)$  is a general nonlinear function of the same form as  $f(\cdot)$  in Equation 11, but with the inclusion of appropriate time-scaling constants. The minimization of this general objective function becomes a parameter optimization problem, since  $\vec{y}(t)$  is a function of the  $3P(n-1)$  free knot derivatives. This reduction of the minimization problem to a parameter optimization problem is similar to approaches used before [11, 12]. But these previous approaches are applicable only for path planning between an initial point and a final point, and do not allow for intermediate knots. Two specific examples of optimization (minimum jerk and minimum energy) are considered in the following subsections.

## 5.1 Minimum Jerk

Suppose that the user desires to minimize the jerk of each joint throughout its trajectory. The joint position errors of the path tracker increase with the magnitude of jerk [6]. In addition, there is evidence that the human brain minimizes joint jerk when planning the movement of an arm [2]. So minimizing some function of jerk would seem to be desirable. The objective function of Equation 11 could then be written as

$$J = \int_0^T [\theta'''(t)]^2 dt. \quad (13)$$

Since the jerk of each joint is decoupled from the other joints, the minimization of Equation 13 can be performed one joint at a time. Setting the partial derivatives of Equation 13 to zero, using Equation 12), and noting that  $y_i(t)$  is a function of  $y_j^{(r)}$  only for  $j \in [i-1, i]$ , gives

$$\frac{\partial}{\partial y_i^{(r)}} \left( \int_0^{\pi/4} \{[y_i'''(t)]^2 + [y_{i+1}'''(t)]^2\} dt \right) = 0, \quad (i = 1, \dots, n-1), \quad (r = 1, 2, 3). \quad (14)$$

The  $3(n-1)$  functions above are linear in the 12 parameters  $(\theta_j, y_j^{(r)})$ , ( $j = i-1, i, i+1$ ), ( $r = 1, 2, 3$ ) (see Equations 5,8). So Equation 14 can be reduced to the  $(n-1)$  matrix equations

$$D_0 \begin{pmatrix} \theta_{i-1} \\ \theta_i \\ \theta_{i+1} \end{pmatrix} + \sum_{k=1}^3 D_k Y_{i-2+k}^{(r)} = 0, \quad (i = 1, \dots, n-1) \quad (15)$$

where the  $D_i$  are  $3 \times 3$  matrices, and  $Y_i^{(r)}$  is defined as

$$Y_i^{(r)} \equiv (y_i' \quad y_i'' \quad y_i''')^T. \quad (16)$$

Assuming that the derivatives of the trajectory are constrained at the endpoints, Equation 15 can be written as a single block tridiagonal matrix equation which looks like

$$B_n \begin{pmatrix} Y_1^{(r)} \\ Y_2^{(r)} \\ \vdots \\ Y_{n-1}^{(r)} \end{pmatrix} = \begin{pmatrix} C_1 \\ C_2 \\ \vdots \\ C_{n-1} \end{pmatrix} \quad (17)$$

where the  $3(n-1) \times 3(n-1)$  matrix  $B_n$  is given by

$$B_n = \begin{pmatrix} D_2 & D_3 & 0 & 0 & 0 & \dots & 0 \\ D_1 & D_2 & D_3 & 0 & 0 & \dots & 0 \\ 0 & D_1 & D_2 & D_3 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ 0 & \dots & \dots & 0 & D_1 & D_2 & D_3 \\ 0 & \dots & \dots & \dots & 0 & D_1 & D_2 \end{pmatrix} \quad (18)$$

and the constant  $3 \times 1$   $C_i$  vectors (functions of  $D_0$ ) can be derived from Equation 15. It can be shown that  $B_n$  is always nonsingular. This property follows from the fact that Equation 13 is always greater than zero unless all of the knot angles and derivatives are zero. The 4  $D_i$  matrices are found to be as follows.

$$\begin{aligned} D_0 &= \begin{pmatrix} 3048.238 & -1690666 & -3048.238 \\ -429.3317 & 858.6634 & -429.3317 \\ 11.91052 & -0.097204 & -11.91052 \end{pmatrix} \\ D_1 &= \begin{pmatrix} 1124.790 & 140.2255 & 3.508263 \\ -140.2255 & -15.34489 & -0.275880 \\ 3.508263 & 0.275880 & -0.011973 \end{pmatrix} \\ D_2 &= \begin{pmatrix} 2538.587 & -0.036400 & 11.69254 \\ -0.036400 & 75.25705 & -0.000894 \\ 11.69254 & -0.000894 & 0.216714 \end{pmatrix} \\ D_3 &= D_1^T \end{aligned} \quad (19)$$

Of course, the same procedure as that outlined in this section can be used to find  $D_i$  matrices which could be used to minimize some user-specified combination of velocity, acceleration, and jerk for each joint. Then the objective function of Equation 13 would be replaced with

$$J = \int_0^T \{k_1[\theta'(t)]^2 + k_2[\theta''(t)]^2 + k_3[\theta'''(t)]^2\} dt \quad (20)$$

where  $k_1$ ,  $k_2$ , and  $k_3$  are user-specified weights. The optimum interior knot derivatives would still be given by Equation 17; and the  $D_i$  matrices would still exhibit the symmetries seen in Equations 19, but would be functions of  $k_1$ ,  $k_2$ , and  $k_3$ .

## 5.2 Minimum Energy

Recall that the  $P$ -element torque vector of a  $P$ -joint robot can be given as

$$T(\vec{\theta}) = \mathcal{M}(\vec{\theta})\vec{\theta}' + \mathcal{S}(\vec{\theta}, \vec{\theta}'), \quad (21)$$

where  $\mathcal{M}$  is the  $P \times P$  mass matrix, and  $\mathcal{S}$  is a  $P$ -vector of centrifugal, Coriolis, and gravity terms. Suppose the user wants to choose the interior knot derivatives of the trigonometric spline for each joint so as to minimize torque. Then the objective function could be written as

$$J = \int_0^T T^T(\vec{\theta})R T(\vec{\theta}) d\tau \quad (22)$$

where  $R$  is a  $P \times P$  positive-definite weighting matrix. The torque vector  $T$  can be written as a function of the normalized joint trajectories as follows:

$$T(\vec{y}) = M(\vec{y})(n\pi/4T)^2 \vec{y}'' + S(\vec{y}, (n\pi/4T)\vec{y}'). \quad (23)$$

This objective function could also be minimized with respect to  $T$  (the actual path length). Some numerical method can be used to find the minimum of Equation 22, and thus determine the interior knot derivatives which would yield a (locally) minimum-energy trigonometric spline.

Knot	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 1: Knot Angles

## 6. Numerical Examples

Software has been written to implement the Trajectory Formulation Algorithm given in Section 4. The first three derivatives of the trajectory at the endpoints were chosen to be zero. The derivatives of the normalized-time trajectory (i.e. each spline segment having a  $(\pi/4)$ -second duration) at the interior knots were calculated using the following heuristic algorithm.

$$\begin{aligned}
 y'_i &= (\theta_{i+1} - \theta_{i-1})/(\pi/2). \\
 y''_i &= 2(\theta_{i+1} - 2\theta_i + \theta_{i-1})/(\pi/4)^2. \\
 y'''_i &= 4(y''_{i+1} - y''_{i-1})/(\pi/2).
 \end{aligned} \quad (24)$$

This trajectory is referred to as the *nominal* trigonometric spline.

Software has also been written to calculate the optimal values of the interior knot derivatives to minimize the objective function of Equation 13 by solving Equation 17. This trajectory is referred to as the *minimum jerk* trigonometric spline.

Software has also been written to implement  $2^{nd}$ -order Schoenberg splines. The normalized path length was chosen to be  $\pi$ , and the first three derivatives of the trajectory at the endpoints were chosen to be zero. Schoenberg splines possess an attractive minimal property (see Equations 1 and 2), but the computational effort required is excessive.

Six joint trajectories were calculated with each method. Each joint trajectory has 8 evenly spaced knots, corresponding to the examples given in [7] and [10]. Each path length is 32 seconds. The joint space knot angles are given in Table 1.

Trajectory plots of the nominal trigonometric splines and algebraic quartic splines for joints 2 and 3 are given in Figures 1 and 2. The superiority of the trigonometric splines is apparent.

For 4 different path planning methods, the *maximum* magnitude of each of the first 3 derivatives are calculated for 6 trajectories. The 4 methods are (1) Quartic Splines [10]; (2) Schoenberg Splines; (3)

Joint	Quartic	Schoenberg	Nominal Trigonometric
1	30,21,30	29,10,8	31,13,16
2	30,17,17	20,7,4	22,8,9
3	56,31,23	58,30,20	52,36,46
4	104,63,71	58,31,22	52,28,33
5	54,31,38	36,20,14	30,18,22
6	22,9,12	21,9,4	24,10,11
Averages	49,29,32	37,18,12	35,19,23

Table 2: Max |Velocity|, |Acceleration|, |Jerk|

Joint	Nominal Trigonometric	Minimum Jerk Trigonometric
1	31,13,16	28,10,8
2	22,8,9	20,7,4
3	52,36,46	50,25,21
4	52,28,33	50,28,22
5	30,18,22	29,17,13
6	24,10,11	21,8,5
Averages	35,19,23	33,16,12

Table 3: Max |Velocity|, |Acceleration|, |Jerk|

Nominal Trigonometric Splines; and (4) Minimum Jerk Trigonometric Splines. Table 2 provides a comparison of the first 3 methods, where no parameter optimization is performed. Table 3 shows how the smoothness of the trigonometric spline is improved when minimum jerk optimization is performed. The units of the first 3 derivatives in Tables 2 and 3 are deg/sec, deg/sec<sup>2</sup>, and deg/sec<sup>3</sup>. It should be noted that the quartic splines have nonzero jerk at the endpoints. The Schoenberg splines and trigonometric splines have zero jerk at the endpoints. The number of continuous derivatives is 3 for the quartic and trigonometric splines, and 8 for the Schoenberg splines.

The algebraic quartic splines used for purposes of comparison in this paper are similar to the splines used in [10]. But the details of the formulation differ. So the numbers in Table 1 and the curves in Figures 1 and 2 are somewhat different than in [10]. The quartic spline averages shown in Table 1, however, are only slightly different that the averages obtained in [10].

It is seen that, in general, both Schoenberg splines and trigonometric splines result in much smoother trajectories than quartic splines. This is true in spite of the fact that quartic splines allow nonzero jerks at the endpoints.

## 7. Conclusion

A new method of manipulator path planning has been presented, and an algorithm has been given. This new method consists of the use of trigonometric splines of order 4. Continuity of the first 3 derivatives is ensured, and the first 3 derivatives at the endpoints can be constrained to any desired values. Trigonometric splines are computationally less expensive than algebraic splines, because there is no need to solve a set of simultaneous linear equations. The derivatives of a trigonometric spline are generally much lower than the derivatives of the corresponding algebraic spline. Trigonometric polynomials are very smooth functions due to the orthogonality of cosine and sine terms. But algebraic splines may oscillate wildly between knots if the order of the spline is too high. Trigonometric splines are also attractive from the standpoint of obstacle avoidance, because each spline segment is dependent only on 2 knots.

## References

- [1] J. Craig, *Introduction to Robotics*. Reading, MA: Addison-Wesley, 1989.
- [2] J. Flanagan and D. Ostry, "Trajectories of Human Multi-Joint Arm Movements . . .," in *Experimental Robotics I*, ed. by V. Hayward and O. Khatib, New York: Springer-Verlag, 1990.
- [3] P. Henrici, *Essentials of Numerical Analysis*. New York: John Wiley & Sons, 1982.
- [4] P. Koch, "Error Bounds for Interpolation by Fourth Order Trigonometric Splines," in *Approximation Theory and Spline Functions*, ed. by Singh *et al.*, pp. 349-360, 1984.
- [5] P. Koch and T. Lyche, "Bounds for the Error in Trigonometric Hermite Interpolation," in *Quantitative Approximation*, ed. by Devore and Scherer, New York: Academic Press, 1980.
- [6] K. Kyriakopoulos and G. Saridis, "Minimum Jerk Path Generation," *IEEE Int. Conf. Rob. Auto.*, vol. 1, pp. 364-369, 1988.
- [7] C. Lin, P. Chang, and J. Luh, "Formulation and Optimization of Cubic Polynomial Joint Trajectories for Industrial Robots," *IEEE Trans. Auto. Control*, vol. AC-28, pp. 1066-1073, Dec. 1983.
- [8] T. Lyche and R. Winther, "A Stable Recurrence Relation for Trigonometric B-Splines," *J. of Approx. Theory*, vol. 25, pp. 266-279, 1979.
- [9] I. Schoenberg, "On Trigonometric Spline Interpolation," *Journal of Mathematics and Mechanics*, vol. 13, pp. 795-825, 1964.

- [10] S. Thompson and R. Patel, "Formulation of Joint Trajectories for Industrial Robots using B-Splines," *IEEE Trans. Ind. Elec.*, vol. IE-34, pp. 192-199, May 1987.
- [11] J. Vlassenbroeck and R. Van Dooren, "A Chebyshev Technique for Solving Nonlinear Optimal Control Problems," *IEEE Trans. Auto. Control*, vol. AC-33, April 1988, pp. 333-340.
- [12] V. Yen and M. Nagurka, "Fourier-Based Optimal Control Approach for Structural Systems," *Journal of Guidance, Control, and Dynamics*, vol. 13, pp. 265-276, March 1990.

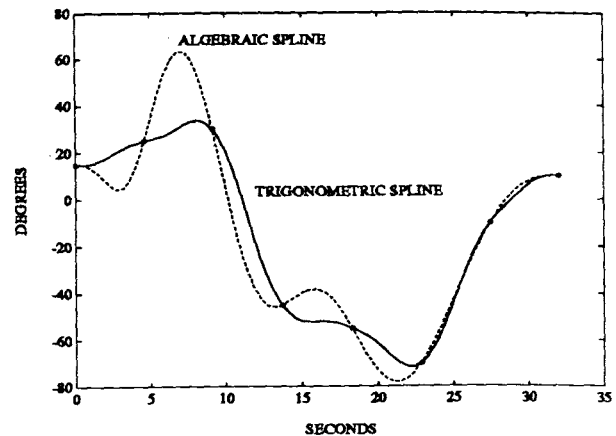


Figure 1: Splines for Joint 2

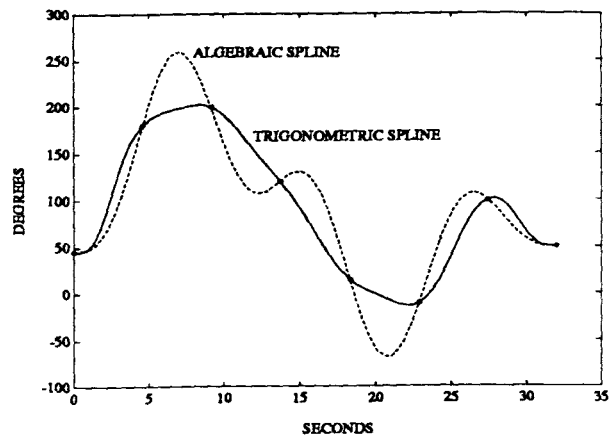


Figure 2: Splines for Joint 3