MODELING, IDENTIFICATION AND CONTROL, 1988, VOL. 9, NO. 2, 69–80
doi:10.4173/mic.1988.2.2

The introduction of *B*-splines to trajectory planning for robot manipulators

PER ERIK KOCH† and KESHENG WANG‡

Keywords: *robotics, trajectory planning,* B-*splines.*

This paper describes how *B*-splines can be used to construct joint trajectories for robot manipulators. The motion is specified by a sequence of Cartesian knots, i.e., positions and orientations of the end effector of a robot manipulator. For a six joint robot manipulator, these Cartesian knots are transformed into six sets of joint variables, with each set corresponding to a joint. Splines, represented as linear combinations of *B*-splines, are used to fit the sequence of joint variables for each of the six joints. A computationally very simple, recurrence formula is used to generate the *B*-splines. This approach is used for the first time to establish the mathematical model of trajectory generation for robot manipulators, and offers flexibility, computational efficiency, and a compact representation.

1. Introduction

In applications such as welding and painting, it is necessary for the robot manipulator to follow the shape of the object on which it is working. In other applications, it may be necessary for the robot manipulator to avoid obstacles between goal points. This motion is specified by a sequence of Cartesian knots, i.e., positions and orientations of the end effector of a robot manipulator.

There are two basic modes of specifying point to point robot motion. One motion strategy is called joint interpolation motion. In joint interpolated motion, trajectories for each joint are planned independently of the end effector motion. All joints start and stop at the same time. In this case, the end effector passes through the two points, but there is no control over its trajectory between the points. A second way of specifying robot motion is to define the end effector trajectory between points. Controlling end effector motion is of course more complex than simple joint interpolated motion. The trajectory of each joint must be controlled to result in the desired trajectory of the end effector.

An end effector move is executed by transforming the end effector position and orientation into joint coordinates at a large number of points on the end effector trajectory. For a 6-joint robot manipulator, these Cartesian knots are transformed into six sets of joint variables, with each set corresponding to a joint.

The use of polynomial functions for generating smooth robot manipulator trajectories is a well-known technique. A plot of joint displacement versus time for a particular move is shown in Fig. 1. The function of approximation for a particular joint must pass through the value calculated for that joint at each of the Cartesian knots. In addition, the function must be continuous in position, velocity, acceleration, even jerk, in order to remain within the physical limitations of the robot manipulator.

Received 29 September 1987.

† Department of Numerical Mathematics, 7034 Trondheim-NTH, Norway.
‡ Production Engineering Laboratory, 7034 Trondheim NTH-SINTEF, Norway.

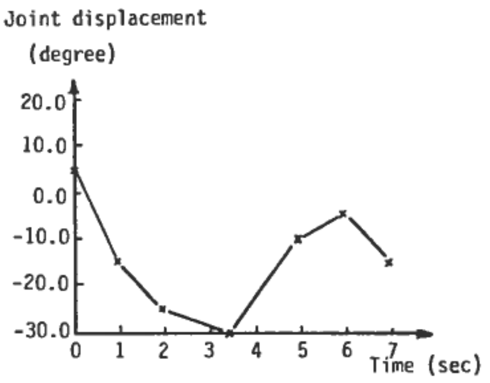


Figure 1. A plot of displacement versus time in a joint.

There are many methods to develop the interpolation polynomials to meet the above physical constraint conditions.

1. A single polynomial passing through all the knot points of joint trajectory results in extremely high degree polynomials. For instance, if the joint trajectory has n knot points, the degree of one polynomial passing through all the points is usually $n - 1$.
2. Piecewise linear interpolation, i.e. a first-degree polynomial (straight line) is used for interpolation between successive knot points. This form of interpolation can be unsatisfactory for a smooth servo response because a linear function in displacement produces impulsive accelerations between constant velocity segments.
3. A number of recent papers have described the application of spline function to the generation of smooth robot joint trajectories (Bollinger and Duffie 1979; Cook and Ho 1982; Luh, Lin and Chang 1983). Splines are used for point to point joint motion and for motion involving a large number of joint knot points. A classification of schemes using splines for constructing a joint trajectory with a large number of knot points is as follows:
   (1) 4-3-3 … 3-4 trajectory
   The first segment is a fourth-degree polynomial specifying the trajectory from the initial position to the left off position of the joint. The subsequent polynomials are all cubics except for the last trajectory segment, which is quartic.
   (2) 3-3-3 … 3 trajectory
   This is a cubic spline fit for all the trajectory segments. The only difference from the 4-3-3 … 3-4 scheme is that the acceleration constraints at the start and destination points are ignored so as to yield only cubic polynomials.
4. Several methods of polynomial curve fitting are described elsewhere (Paul 1981, Craig 1986, Wang and Lien 1987a).

In this paper, we describe a scheme for generating *B*-spline joint trajectories which is widely used in the field of computer graphics for connecting data points with smooth curves (Newman and Sproull 1983). These curves known as *B*-spline functions, have been thoroughly studied and investigated and found to provide good inherent behaviour, such as preventing the loss of accuracy due to cancellation, greatly reducing the amount of computation, helping the convergence analysis (Gordon & Riesenfeld 1974, de Boor 1978, Powell 1981). We also show that if this approach is used to establish the mathematical model of joint trajectory generation for robot manipulators, it will offer flexibility, computational efficiency, and a compact representation.

2. Interpolation by splines

We will use *B*-splines as building blocks to construct curves that interpolate to given points, see e.g. de Boor (1978). *B*-splines of order k are piecewise polynomials of degree $k-1$ in one variable. They are $k-2$ times continuously differentiable, positive on exactly k consecutive intervals and zero everywhere else. Inside their support they are bellshaped.

Let $t_1 \leqslant t_2 \leqslant \ldots \leqslant t_{n+k}$ be a non-decreasing sequence of real numbers, where $n \geqslant k \geqslant 1$. The points $t_1, \ldots, t_{n+k}$ are called knots. In 1972 both de Boor (1972) and Cox (1972) discovered a stable recurrence formula for computing *B*-spline values.

The simplest *B*-spline, that of order 1, is given by

$$B_{i,1}(t) = \begin{cases} 1, & t_i \leqslant t < t_{i+1} \\ 0, & \text{otherwise} \end{cases} \tag{1}$$

The *B*-splines of higher order are recursively determined by

$$B_{i,j}(t) = \frac{t - t_i}{t_{i+j-1} - t_i} B_{i,j-1}(t) + \frac{t_{i+j} - t}{t_{i+j} - t_{i+1}} B_{i+1,j-1}(t)$$
$$i = 1, 2, \ldots, n \qquad j \geqslant 2 \tag{2}$$

The derivative of a *B*-spline is given by the following formula (de Boor 1978)

$$B'_{i,j}(t) = (j-1)\left(\frac{B_{i,j-1}(t)}{t_{i+j-1} - t_i} - \frac{B_{i+1,j-1}(t)}{t_{i+j} - t_{i+1}}\right) \tag{3}$$

A spline of order k is a piecewise polynomial of degree less than k which is $k-2$ times continuously differentiable. Schoenberg (1946) originally introduced the splines and showed together with Curry (1966), that on (t_k, t_{n+1}) splines and linear combinations of *B*-splines are the same thing. The 'B' in *B*-splines stands for basis. The *B*-splines constitute a basis for the vector space of kth order splines.

If $I = [a, b]$ is the global parameter-domain of interest, then by choosing

$$t_1 = t_2 = \ldots = t_k = a, \qquad t_{n+1} = t_{n+2} = \ldots = t_{n+k} = b \tag{4}$$

any spline of order k on I is equal to some linear combination of *B*-splines on the whole of I.

Let $m > 1$. In our case usually $m = 6$. A parametric spline curve of order k in $\mathbb{R}^m$ can now be defined by

$$s(t) = \sum_{j=1}^{n} c_j \,.\, B_{j,k}(t), \qquad t \text{ in } I \tag{5}$$

where the c_js are vectors in $\mathbb{R}^m$.

Let $\tau_1 \leqslant \tau_2 \leqslant \ldots \leqslant \tau_n$ be n abscissas in I and set $d_i = \underset{\tau_{i-j} = \tau_i}{\text{Max}}\, j$.

If $P_1, P_2, \ldots, P_n$ are n points in $\mathbb{R}^m$, then we say that the curve (5) Hermite interpolates to these points at $\tau_1, \tau_2, \ldots, \tau_n$ respectively if

$$s^{(d_i)}(\tau_i) = P_i, \qquad i = 1, 2, \ldots, n \tag{6}$$

Let us relate this to ordinary scalar spline Hermite interpolation. Let $P_{l,i}$ be the lth component P_i and $c_{l,i}$ similar for c_i. By (5), the lth component of (6) is then equal to

$$\sum_{j=1}^{n} c_{l,j} B_{j,k}^{(d_i)}(\tau_i) = P_{l,i}, \qquad i = 1, 2, \ldots, n \tag{7}$$

This is an ordinary linear system of n equations in n unknowns $c_{l,1}, c_{l,2}, \ldots, c_{l,n}$. Hence the interpolation condition (6) is equivalent to m different systems of linear equations. The system (7) has a unique solution if the matrix

$$M = (B_{j,k}^{(d_i)}(\tau_i))_{i=1,\, j=1}^{n,\, n} \tag{8}$$

is non-singular. This is the case if there is at least one interpolation point in each (t_i, t_{i+k}), $i = 1, 2, \ldots, n$ (Schoenberg and Whitney, 1953).

The curve (5) which solves (6) can now be constructed in the following steps:

Step 1. Construct the matrix (8) by using the formulae (2) and (3).

Step 2. Solve the linear system (7) for each $l = 1, \ldots, m$.

Step 3. For each evaluation point t calculate the right side of (5) by using the recurrence relation (2).

If there are many ts for which the sum (5) has to be calculated, it is more economic to first find the different polynomial pieces of each component of s and then use Horner's scheme. This then replaces Step 3 above.

In practice only the vectors $P_1, P_2, \ldots, P_n$ are given so that the interval I, the knots $t_1, t_2, \ldots, t_{n+k}$ and the interpolation points $\tau_1, \tau_2, \ldots, \tau_n$ can be chosen by the user. The first k and last k knots have to be chosen according to (4), and Schoenberg–Whitney condition has to be fulfilled. Since the interpolation points are ordered in an increasing sequence, the following inequalities must hold:

$$t_i < \tau_i < t_{i+k}, \qquad i = 1, 2, \ldots, n \tag{9}$$

where the left inequality may be weakened if k knots coalesce. This leaves quite a bit of freedom. If we want the abscissa τ_i to have multiplicity d, i.e. $\tau_{i-1} < \tau_i = \ldots = \tau_{i+d-1} < \tau_{i+d}$, the difference $\tau_{i+d} - \tau_i$ and the distance between P_i and P_{i+d} ought to be proportional. A reasonable means of selecting the knots is the following (de Boor 1978):

$$t_i = (\tau_{i-1} + \tau_{i-2} + \ldots + \tau_{i-k+1})/(k-1) \qquad i = k+1, \ldots, n. \tag{10}$$

In the next section we will look at some numerical examples for robot manipulators.

3. Numerical examples

In order to illustrate how to use B-splines to construct joint trajectories for robot manipulators, a PUMA 600 robot manipulator with six joints is considered as a numerical example.

Knot joint	1	2	3	4	5	6	7	8	9	10
1	15·0	30·0	50·0	90·0	130·0	90·0	45·0	−10·0	−30·0	−50·0
2	10·0	25·0	30·0	15·0	−20·0	−55·0	−70·0	−20·0	0·0	10·0
3	50·0	70·0	150·0	200·0	120·0	35·0	−10·0	50·0	60·0	50·0
4	15·0	20·0	40·0	80·0	80·0	40·0	−60·0	−100·0	−60·0	−30·0
5	10·0	30·0	10·0	−40·0	−60·0	10·0	50·0	−40·0	−20·0	10·0
6	6·0	20·0	40·0	80·0	70·0	10·0	−10·0	15·0	30·0	20·0

Table 1. Joint variables for the PUMA 600 robot manipulator.

Ten knot points from a Cartesian path of the end effector of the robot manipulator are chosen. Using the closed form solution for the inverse kinematics (Wang and Lien 1987b), the joint variables are solved for these knots and shown in Table 1.

The robot is at rest at the starting point, and comes to a full stop at the end point.

The velocity, acceleration and jerk constraints are given in Table 2.

Since the velocity and acceleration have to be zero at the start and end points, the spline to be constructed must Hermite interpolate to the following points:

$$
\begin{aligned}
P_1 &= (\ 15{\cdot}0,\ 10{\cdot}0,\ 50{\cdot}0,\ 15{\cdot}0,\ 10{\cdot}0,\ 6{\cdot}0)^{\mathrm{T}}\\
P_2 &= (\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0)^{\mathrm{T}}\\
P_3 &= (\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0)^{\mathrm{T}}\\
P_4 &= (\ 30{\cdot}0,\ 25{\cdot}0,\ 70{\cdot}0,\ 20{\cdot}0,\ 30{\cdot}0,\ 20{\cdot}0)^{\mathrm{T}}\\
&\ \ \vdots\\
P_{12} &= (-50{\cdot}0,\ 10{\cdot}0,\ 50{\cdot}0,\ -30{\cdot}0,\ 10{\cdot}0,\ 20{\cdot}0)^{\mathrm{T}}\\
P_{13} &= (\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0)^{\mathrm{T}}\\
P_{14} &= (\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0,\ 0{\cdot}0)^{\mathrm{T}}
\end{aligned}
$$

where $\tau_1 = \tau_2 = \tau_3$ and $\tau_{12} = \tau_{13} = \tau_{14}$ and the rest of the τs are simple ($d_i = 0$).

We want the spline curve to be three times continuously differentiable. Hence we cannot use cubic splines, $k = 4$, since they are only twice continuously differentiable. But quartic ($k = 5$), quintic ($k = 6$) and higher degree splines can be used. We choose quartic splines, since piecewise 4th degree polynomials are cheaper to evaluate, needing only 4 multiplications per point.

We want to find interpolation abscissas and knots (the dividing points between subintervals) on an interval $[0, T]$, where the spline curve Hermite interpolates to the data and satisfies all the joint constraints, and where the total time T is as short as possible.

Joint constraint	1	2	3	4	5	6
Velocity (degree/sec)	100·0	95·0	100·0	150·0	130·0	110·0
Acceleration (degree/sec^2)	45·0	40·0	75·0	70·0	90·0	80·0
Jerk (degree/sec^3)	60·0	60·0	55·0	70·0	75·0	70·0

Table 2 Joint constraints of the PUMA 600 robot manipulator.

Let us observe that we can work with a fixed interval, for instance [0, 20], and scale this interval at the end of the calculations.

If we find interpolation abscissas τ_i and knots t_i on [0, 20] for which all the constraints are satisfied, that means we can use a linear scaling to a smaller interval $[0, T]$. The new abscissas and knots will be $(T/20)\tau_i$ and $(T/20)t_i$.

If s_j was the joint j spline on [0, 20], then the corresponding scaled spline on $[0, T]$ will be $s_j\,((20/T)t)$. Let us denote the constraint for joint j and the ith derivative by M_{ji}. By differentiation, the following inequalities have to hold

$$\left(\frac{20}{T}\right)^i \| s_j^{(i)} \|_\infty \leqslant M_{ji}, \qquad \begin{array}{l} i = 1, 2, 3 \\ j = 1, 2, \ldots, 6 \end{array} \tag{11}$$

where $\| \ \|_\infty$ denotes the sup-norm. By (11), the least total time T we can have and still satisfy all the constraints, is the following

$$T = 20 \underset{1 \leqslant j \leqslant 6}{\text{Max}} \ \underset{1 \leqslant i \leqslant 3}{\text{Max}} \left(\frac{\|s_j^{(i)}\|_\infty}{M_{ji}}\right)^{1/i} \tag{12}$$

This T is a function of the abscissas τ_i and knots t_i on [0, 20]. Here, $\tau_1 = \tau_2 = \tau_3 = 0$ and $\tau_{12} = \tau_{13} = \tau_{14} = 20$ and $n = 14$.

The knots have to satisfy (4), so that $t_1 = t_2 = \ldots = t_5 = 0$ and $t_{15} = t_{16} = \ldots = t_{19} = 20$. Hence T given by (12) is a function of the 8 internal abscissas $\tau_4, \ldots, \tau_{11}$ and the 9 internal knots $t_6, \ldots, t_{14}$.

It seems natural to use some optimization algorithm to minimize T. We chose the conjugate gradient method, (see Rao 1978). Let X denote the vector $(\tau_4, \ldots, \tau_{11}, t_6, \ldots, t_{14})^{\mathrm{T}}$, so that (12) becomes a function of X, $T = T(X)$. We want to find an X that minimizes $T(X)$. The conjugate gradient method can be summarized as follows:

Step 1. Start with a vector $\mathbf{X}_0$ and compute the negative gradient $\mathbf{S}_0 = -\nabla T(X_0)$.

Step 2. Find the point X_1 along the direction S_0 from X_0, $X_1 = X_0 + \lambda_1^* S_0$ so that T becomes a minimum along this line. Set $i = 1$ and proceed.

Step 3. Compute $\nabla T_i = \nabla T(X_i)$ and set $S_i = -\nabla T_i + (\|\nabla T_i\|/\|\nabla T_{i-1}\|)^2 \,.\, S_{i-1}$.

Step 4. Find the point X_{i+1} along the direction S_i from X_i, $X_{i+1} = X_i + \lambda_i^* S_i$ so that T becomes a minimum along this line.

Step 5. If X_{i+1} is optimal, stop. Otherwise set $i = i + 1$ and repeat Step 3, 4 and 5.

We used numerical differentiation to approximate the partial derivatives. The secant method was used to solve the one dimensional minimization problem, i.e. to find the minimum of a function $g(t)$ of only one real variable t.

For each abscissa and knot set the six different interpolating splines were constructed and transformed to piecewise polynomial form by some subroutines written by Carl de Boor and modified by D. E. Amos, see the SLATEC library of FORTRAN subroutines and (de Boor 1977).

Our FORTRAN program needs a set of starting values for abscissas and knots. Let us now only consider those vectors P_i that correspond to spline values and not to derivatives. Set $P_1^* = P_1$, $P_2^* = P_4$, $P_3^* = P_5, \ldots, P_{10}^* = P_{12}$ and $\tau_1^* = \tau_1$, $\tau_2^* = \tau_4$, $\ldots$, $\tau_{10}^* = \tau_{12}$. It seemed reasonable that the difference between two abscissas,

Experiment no.	Weight W_1	Total time T (sec)
1	1·0	33·32
2	2·0	18·89
3	3·0	17·81
4	3·5	18·06
5	2·7	17·50
6	2·5	17·21
7	2·3	16·85

Table 3. Total time after one iteration.

$\tau_{i+1}^* - \tau_i^*$, should be proportional to the distance between P_i^* and P_{i+1}^*, $\|P_{i+1}^* - P_i^*\|_\infty$. We tried with a weighted proportionality,

$$\tau_{i+1}^* - \tau_i^* = 20 \frac{W_i \|P_{i+1}^* - P_i^*\|_\infty}{\sum_{j=1}^{9} W_j \|P_{j+1}^* - P_j^*\|_\infty}, \qquad i = 1, 2, \ldots, 9 \tag{13}$$

where the Ws are some chosen positive weights. It is reasonable to have higher weights at the end points since the robot manipulator starts and stops with zero velocity and acceleration. Therefore it must have more time there. So let $W_2 = W_3 = \ldots = W_8 = 1$ and $W_1 = W_9$. The knots were then chosen according to (10) in the start.

We tried one iteration of the conjugate gradient method for some different W_1s. Only one iteration of this method really means a search for a minimum of T along the negative gradient of T (i.e. one iteration with the steepest descent method). Table 3 shows the results for some choices of W_1.

We notice from Table 3 that the last experiment where $W_1 = 2{\cdot}3$ gave the least total time after one iteration with the conjugate gradient method. The interpolation abscissas and knots on [0, 20] in this case are shown in Table 4.

The plots are given in Fig. 2. For each of the six joints the sup-norms of the velocity, acceleration and jerks are given by Table 5.

i	τ_i	t_i
1	0·000	0·000
2	0·000	0·000
3	0·000	0·000
4	2·186	0·000
5	3·868	0·000
6	5·508	1·468
7	8·006	2·748
8	10·659	4·731
9	13·780	7·028
10	16·590	9·486
11	17·840	12·258
12	20·000	14·718
13	20·000	17·052
14	20·000	18·607

Table 4. Interpolation points and knots for the case $W_1 = 2{\cdot}3$ after 1 iteration.

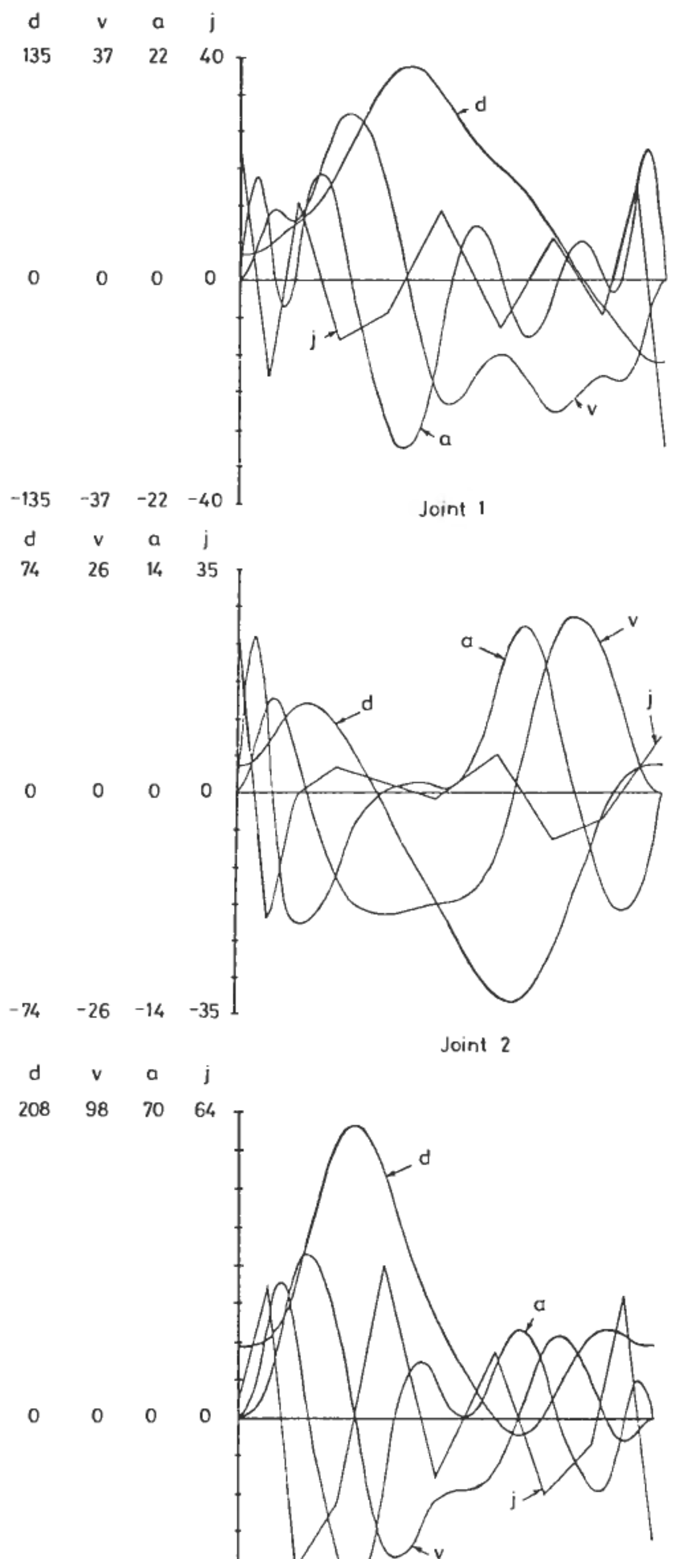
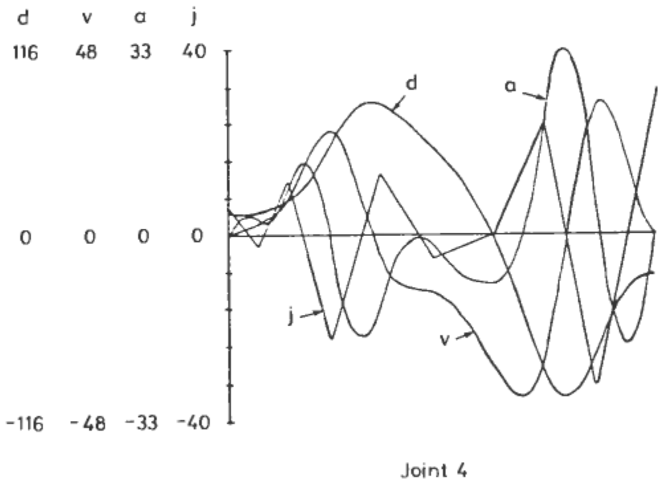
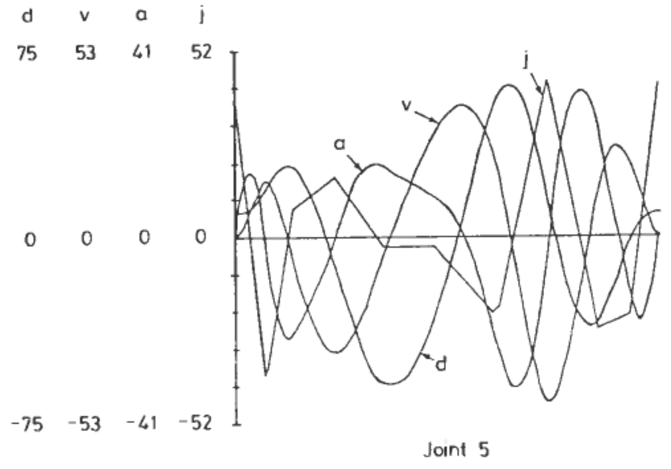


Figure 2. Joint trajectories for the example; d = displacement, v = velocity, a = acceleration, j = jerk.





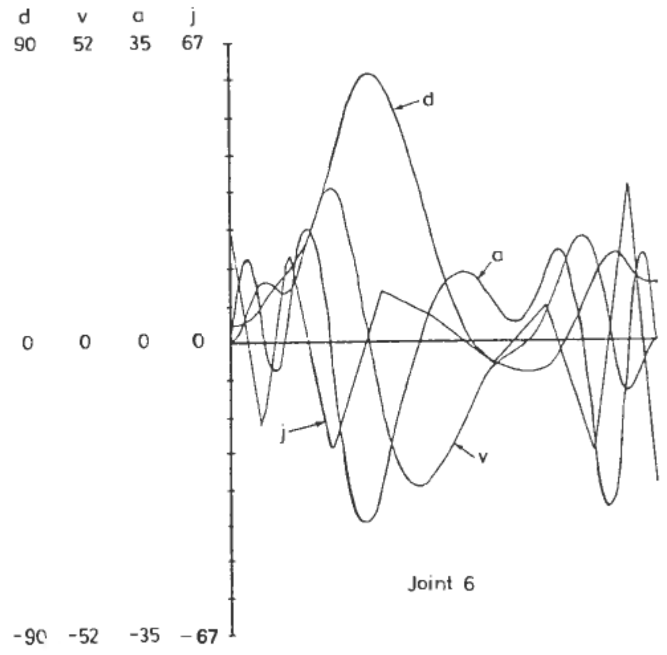


Figure 2.—*continued*

Joint	Velocity	Acceleration	Jerk
1	27·3	16·2	29·7
2	20·9	10·9	24·6
3	52·9	38·2	31·8
4	40·9	33·1	32·2
5	46·3	35·5	44·0
6	26·8	21·5	35·1

Table 5. The maximum value of the joint variables.

Next we tried to run the program with different weights W_1 in the start and with 6 iterations of the conjugate gradient method. The result is shown in Table 6.

Experiment no.	Weight W_1	Total time T (sec)
1	1·0	19·69
2	2·0	17·70
3	3·0	16·30
4	3·5	17·95
5	2·7	17·28
6	2·5	16·79
7	2·3	16·77

Table 6. The total time after 6 iterations.

We notice from Table 6 that it is an advantage to begin with the weight $W_1 = 3{\cdot}0$ even though $W_1 = 2{\cdot}3$ was favourable in the one-iteration case. Hence we chose the weight $W_1 = 3{\cdot}0$. But even after more than 10 iterations the total time had only crept down to 16·22. A local minimum had been reached. We retreated to the situation just after iteration number 6, restarted the program with a slightly different set of abscissas and knots, and performed 7 iterations of the conjugate gradient method. This yielded $T = 15{\cdot}35$. Then a slight perturbation of the parameter-vector, restart, and 4 iterations gave $T = 15{\cdot}10$. After some searching for a good restart vector and 5 iterations we obtained $T = 14{\cdot}80$. The abscissas and knots for this final stage are given in Table 7 and the sup-norms for every derivative of the six joints are given in Table 8.

i	τ_i	t_i
1	0·000	0·000
2	0·000	0·000
3	0·000	0·000
4	2·420	0·000
5	4·214	0·000
6	5·647	1·610
7	8·499	2·957
8	10·117	4·781
9	13·008	7·001
10	16·101	9·484
11	17·613	11·943
12	20·000	14·502
13	20·000	16·902
14	20·000	18·274

Table 7. Interpolation points and knots after a total of 22 iterations.

Joint	Velocity	Acceleration	Jerk
1	31·3	23·3	22·4
2	22·7	12·1	16·4
3	55·7	35·6	22·2
4	36·4	23·6	25·9
5	47·8	29·9	30·4
6	39·4	24·2	24·7

Table 8. The maximum value of the joint variables.

From Table 8 and Table 2, we notice that the sup-norm for the jerk of joint 5 has the same relation to its joint constraint, viz. $30{\cdot}4/75 = 0{\cdot}405$, as that of the sup-norm for the jerk of joint 3, viz. $22{\cdot}2/55 = 0{\cdot}404$. In all the previous experience the jerk of joint 5 has given the maximum value in (12), so that we probably have reached a global minimum for T. In any case, to lower T further would now be much more difficult. Some experimentations showed this. After many iterations and restarts, we only reached $T = 14{\cdot}71$.

4. Conclusions

Computer programs have been written to implement the procedure for *B*-splines joint trajectory for robot manipulators. The new approach makes velocity, acceleration, and jerk all continuous. The continuous jerk is a very important parameter for the motion of robot manipulators. To obtain continuity in the jerk it is necessary to work with splines with a higher degree than 3. Since the cubic splines approach could not make the jerk continuous, we have worked with quartic splines. The problem with these quartic splines was solved by using the *B*-spline representation. The output to the robot manipulator is the knots and the coefficients for the polynomials that constitute the spline on each interval. For each point, 4 multiplications are necessary for evaluation.

This approach also offers more flexibility than was previously possible, it is easy to switch from Lagrange to Hermite interpolation incorporating zero velocity and acceleration at the starting and end points. No ad hoc scheme is needed. Since knots and interpolation abscissas need not coincide, more freedom is added in searching for spline curves that satisfy all the constraints for a small time T.

The conjugate gradient method has been used in the computer program in order to minimize the total time T given by (12). This algorithm often converges to local minima depending upon the input parameter vector. Therefore, when two iterations give the same T the program gives the control to the user, so that the user interactively can change components of the parameter vector (abscissas and knots). The user may then decide when to start the conjugate gradient algorithm again. After the user is satisfied the program furnishes the scaled interval [0, 20], the scaled abscissas and knots and every derivative of each of the polynomial pieces for each joint. This is the information needed to be stored and retrieved by the controller of the robot manipulator at run time.

References

Bollinger, J., and Duffie, N. (1979), Computer algorithms for high speed continuous path robot manipulator, *Ann. CIRP* **28**, 391–395.

COOK, C. C., and HO, C. Y. (1982), The application of spline functions to trajectory generation for computer-controlled manipulators, *Digital Systems for Industrial Automation*, **1,** 325–333.

COX, M. G. (1972), The numerical evaluation of *B*-splines, *J. Inst. Math. Applic.*, **10,** 134–149.

CRAIG, J. J. (1986), *Introduction to robotics mechanics and control* (Addison-Wesley Publishing Company) pp. 191–219.

CURRY, H. B., and SCHOENBERG, I. J. (1966), On polya frequency functions IV: The fundamental spline functions and their limits. *J. d'Analyse Math.* **17,** 71–107.

DE BOOR, C. (1972), On calculating with *B*-splines, *J. Approximation Theory*, **6,** 50–62.

DE BOOR, C. (1977), Package for calculating with *B*-splines, *SIAM Journal of Numerical Analysis* **14,** 441–472.

DE BOOR, C. (1978), *A practical guide to splines*, (New York: Springer Verlag) pp. 219 & 138.

GORDON, W. J., and RIESENFELD, R. F. (1974), *B*-spline curves and surfaces in *Computer aided geometric design*, edited by Barnhill, R. E. and Riesenfeld, R. F., (Academic Press), 95–126.

LUH, J. Y. S., LIN, C. S., and CHANG, P. R. (1983), Formulation and optimization of cubic polynomial joint trajectory for industrial robots, *IEEE Trans. Automatic Control*, **28,** 1066–1074.

NEWMAN, W. M., and SPROULL, R. F. (1983), *Principles of interactive computer graphics*, McGraw-Hill Inc) pp. 320–325.

PAUL, R. C. (1981), *Robot manipulators: mathematics, programming and control*, (Cambridge: MIT Press), 119–155.

POWELL, M. J. D. (1981), *Approximation theory and methods* Cambridge University Press) pp. 227–236.

RAO, S. S. (1978), *Optimization Theory and Applications*, (Wiley Eastern Limited) pp. 307.

SCHOENBERG, I. J. (1946), Contribution to the problem of approximation of equidistant data by analytic functions, *Quart. Appl. math.*, **4,** 45–49, 112–141.

SCHOENBERG, I. J., and WHITNEY, A. (1953), On polya frequency functions III: The positivity of translation determinant with applications to the interpolation problem by spline curves. *Trans. Amer. Math. Soc.*, **74,** 246–259.

WANG, K., and LIEN, T. K. (1987a), The planning of straight line trajectory in robotics using interactive computer graphics, *Modeling, Identification and Control*, **8,** 125–135.

WANG, K., and LIEN, T. K. (1987b), The solution with closed form for the inverse kinematics of PUMA robot manipulator, Proceedings of the International Conference on The Robotics, Paper No. 10, Yugoslavia.