

## G2.5 Neural networks for optimal robot trajectory planning

*Dan Simon*

### Abstract

This case study discusses the interpolation of minimum-jerk robot joint trajectories through an arbitrary number of knots using a hard-wired neural network. Minimum-jerk joint trajectories are desirable for their similarity to human joint movements and their amenability to accurate tracking. The resultant trajectories are numerical functions of time. The interpolation problem is formulated as a constrained quadratic minimization problem over a continuous joint angle domain and a discrete time domain. Time is discretized according to the robot controller rate. The outputs of the neural network specify the joint angles (one neuron for each discrete value of time) and the Lagrange multipliers (one neuron for each trajectory constraint). An annealing method is used to prevent the network from getting stuck in a local minimum. We show via simulation that this trajectory planning method can be used to improve the performance of other trajectory optimization schemes.

### G2.5.1 Project overview

#### G2.5.1.1 Robot trajectory planning

The industrial robot is a highly nonlinear, coupled multivariable system with nonlinear constraints. For this reason, robot control algorithms are often divided into two stages: *path planning* and *path tracking* (Craig 1989). Path planning is often done without much consideration for the robot dynamics, and with simplified constraints. This reduces the computational expense of the path planning algorithm. The output of the path planning algorithm is then input to a path tracking algorithm.

There are algorithms for the robot control problem which do not separate path planning and path tracking. These algorithms take source and destination Cartesian points as inputs, and determine optimal joint torques. Shiller and Dubowsky (1989) provide a concise review of such algorithms. While such methods are attractive in that they provide optimal solutions to some robot control problems, they result in impractically complicated algorithms and a large computational expense. A simpler approach to the robot control problem is to generate a suboptimal joint trajectory, and then track the trajectory with a controller. This approach ignores most of the dynamics of the robot. So the resultant trajectories do not take full advantage of the robot's capabilities, but are computationally much easier to obtain. In this approach, a number of knot points are chosen along the desired Cartesian path. The number of knots chosen is a tradeoff between exactness and computational expense. The Cartesian knots are then mapped into joint knots using inverse kinematics. Finally, for each robot joint, an analytic interpolating curve is fit to the joint knots. Some of the initial and final derivatives of the curve are constrained to zero so as to ensure that the robot begins and ends its motion smoothly. 'Smoothness' is a concept which combines the ideas of derivative continuity and derivative magnitudes.

The most popular type of interpolation is algebraic splines (Lin and Chang 1983, Lin *et al* 1983, Thompson and Patel 1987). Higher-order splines result in continuity of higher-order derivatives, which reduces wear and tear on the robot (Craig 1989) but this is at the expense of large oscillations of the

trajectory. Trigonometric splines can be used to provide a less oscillatory interpolating curve (Simon and Isik 1993).

### G2.5.1.2 Motivation for a neural solution

Consider a sequence of knots through which an interpolating curve is required to pass. A human could create an interpolating curve, but in a different way than a computer algorithm would. Computer algorithms can calculate analytic functions which pass through given knots. A human can draw a smooth curve through a given set of knots, but without performing any mathematical calculations. In contrast with the computer algorithm, the interpolating curve drawn by the human would not be an analytic function of time. In addition, the human would not satisfy the constraints exactly, but only approximately. For example, if the human was requested to maintain a zero slope at the endpoints, the resulting slope would not be zero, but would be very small. Such a result would be satisfactory for most robot path planning applications. These facts indicate that an artificial neural network may be able to do well at interpolation.

Of course, artificial neural networks are still quite far from any biological neural networks. Further motivation for seeking a neural solution to the robot trajectory optimization problem is obtained from the possibility of *implementation in parallel hardware*. This would give the advantage of quick solutions to large problems which would not otherwise be practical using more conventional optimization methods.

The robot path planning problem can be viewed as an optimization problem: given a desired set of knots and endpoint constraints, find the 'best' interpolating curve such that the knot errors and endpoint derivatives are not too 'large'. Several researchers have solved continuous optimization problems using neural networks (Zhao and Mendel 1988, Jeffrey and Rosner 1986a, b, Jang *et al* 1988). Platt and Barr (1988) formulate a neural network which can calculate a minimum of a general function subject to inequality or equality constraints. Their network has the important property of local stability for the problem considered in this section. Due to its stability and generality, this is the network which is used to determine a minimum-jerk robot joint path through a given set of knots.

In order to plan an optimal robot trajectory, the measure of optimality must be defined. Human arm movements satisfy some optimality criterion, and this would seem to be a desirable criterion to adopt when planning trajectories for robot arms. Flash and Hogan (1985) suggest that human arm movements minimize a measure of Cartesian jerk, while Flanagan and Ostry (1990) present evidence that a function of joint jerk is minimized. Uno *et al* (1989) and Kawato *et al* (1990) argue that the objective function is a measure of the derivative of the joint torques, and propose a neural network to learn such a trajectory. In this section, a joint jerk objective function is used. While this choice ignores the dynamics of the robot, it reduces the error of the path tracker (Kyriakopoulos and Saridis 1988) and thus is suitable for robotics applications.

## G2.5.2 Design process

### G2.5.2.1 Topology

Platt and Barr (1988) formulate a neural network which can be used for constrained minimization. Their algorithm, along with some straightforward extensions, is summarized in the following paragraphs.

Consider the following constrained minimization problem:

$$\min f(\mathbf{x}) \text{ subject to } \mathbf{g}(\mathbf{x}) = 0 \quad (\text{G2.5.1})$$

where  $f(\cdot)$  is a scalar functional,  $\mathbf{x}$  is an  $n$ -vector of independent variables, and  $\mathbf{g}(\cdot)$  is a vector-valued function mapping  $\mathcal{R}^n \rightarrow \mathcal{R}^m$ .

Lagrange multipliers can be used to convert the constrained problem of (G2.5.1) into the following unconstrained problem:

$$\min [f(\mathbf{x}) + \boldsymbol{\lambda}^T \mathbf{g}(\mathbf{x})] \quad (\text{G2.5.2})$$

where  $\boldsymbol{\lambda}$  is an  $m$ -vector of Lagrange multipliers associated with the constraints  $\mathbf{g}(\cdot)$ . A necessary condition for the solution of (G2.5.2) is

$$\frac{\partial f}{\partial \mathbf{x}} + \boldsymbol{\lambda}^T \frac{\partial \mathbf{g}}{\partial \mathbf{x}} = 0. \quad (\text{G2.5.3})$$

Now consider a neural network with dynamics of the form

$$\begin{aligned} \dot{x}_i &= -\frac{\partial f}{\partial x_i} - \sum_{\alpha=1}^m (\lambda_{\alpha} + c_{\alpha} g_{\alpha}) \frac{\partial g_{\alpha}}{\partial x_i} \quad (i = 1, \dots, n) \\ \dot{\lambda}_j &= c_j g_j \quad (j = 1, \dots, m) \end{aligned} \tag{G2.5.4}$$

where  $c$  is an  $m$ -vector of constants. Assume that the constraints  $g(\cdot)$  of the original problem (G2.5.1) are linear functions of  $x$ . Then differentiating  $\dot{x}_i$  in (G2.5.4) gives

$$\ddot{x}_i + \sum_{j=1}^n \frac{\partial^2 f}{\partial x_i \partial x_j} \dot{x}_j + \sum_{\alpha=1}^m \left[ c_{\alpha} \left( g_{\alpha} + \sum_{j=1}^n \frac{\partial g_{\alpha}}{\partial x_j} \dot{x}_j \right) \frac{\partial g_{\alpha}}{\partial x_i} \right] = 0. \tag{G2.5.5}$$

Now consider the candidate Lyapunov energy function

$$E = \frac{1}{2} \sum_{i=1}^n (\dot{x}_i)^2 + \frac{1}{2} \sum_{\alpha=1}^m c_{\alpha} g_{\alpha}^2. \tag{G2.5.6}$$

The derivative of this energy function is a quadratic function

$$\dot{E} = -\dot{x}^T A \dot{x}. \tag{G2.5.7}$$

It has been shown in the literature (Platt and Barr 1988, Arrow *et al* 1958) that there exists a finite vector  $c$  such that matrix  $A$  is positive definite at the constrained minima of (G2.5.1). If  $A$  is continuous, then it is positive definite in some region surrounding each constrained minimum. Therefore, if the dynamic system defined by (G2.5.4) begins in that region and remains in that region, the system will settle into the zero-energy state where

$$\dot{x} = 0 \tag{G2.5.8}$$

$$g(x) = 0. \tag{G2.5.9}$$

Now  $g(x) = 0$  implies that the original constraints are satisfied, and  $\dot{x} = 0$  implies (G2.5.4) that

$$\frac{\partial f}{\partial x_i} + \sum_{\alpha=1}^m (\lambda_{\alpha} + c_{\alpha} g_{\alpha}) \frac{\partial g_{\alpha}}{\partial x_i} = 0 \tag{G2.5.10}$$

which satisfies the necessary conditions for a local minimum of the original constrained problem (G2.5.3).

To sum up, equation (G2.5.4), with an appropriately chosen  $c$ , converges to a solution of the original constrained minimization problem of (G2.5.1). Equation (G2.5.4) is in the form of first-order differential equations, which implies that it could be implemented in parallel hardware to yield a very quick solution.

### G2.5.2.2 Development details

When interpolating the path of a robot joint between a set of joint space knots, it is desirable to obtain as smooth a solution as possible. This results in an appearance of coordination (Flanagan and Ostry 1990), reduces wear on the robot joints and prevents the excitation of resonances (Craig 1989), and improves the accuracy of the path tracker (Kyriakopoulos and Saridis 1988). Therefore, in robot trajectory generation, the interpolation problem for each joint can be stated as follows.

Given a set of  $L$  knots for a robot joint, determine a function  $\theta(t)$  which

- is as 'smooth' as possible
- has 'small' errors at the knots
- has 'small' derivatives at the endpoints.

Smoothness can be defined as the integral of the square of the jerk of the position trajectory (Flanagan and Ostry 1990). In order for the robot joint to start and stop its motion in a smooth manner, the first three derivatives at the endpoints should be small. If the path length is  $T$  s, and the desired knot angles

are  $\theta(t_j) = \phi_j$  ( $j = 1, \dots, L$ ), then the optimization problem for each joint can be written as

$$\begin{aligned} & \min \int_0^T [\theta'''(t)]^2 dt && \text{(G2.5.11)} \\ \text{subject to} & \theta(t_j) = \phi_j && (j = 1, \dots, L) \\ & \theta'(0) = 0 \\ & \theta'(T) = 0 \\ & \theta''(0) = 0 \\ & \theta''(T) = 0 \\ & \theta'''(0) = 0 \\ & \theta'''(T) = 0. \end{aligned}$$

If the  $L$  knots are equally spaced in time, then the knot times  $t_i$  satisfy

$$t_i = (i - 1)T / (L - 1) \quad (i = 1, \dots, L). \quad \text{(G2.5.12)}$$

The joint trajectory at the endpoints is exactly constrained. That is, the joint angles at  $t = 0$  and  $t = T$  are fixed constants. But the joint angles at the interior knot times are not truly equality constraints; the interior knot angles are more like centers of tolerance *near* which the joint trajectory is required to pass. Also, the first three endpoint derivatives do not need to be exactly zero. As long as they are very small, the robot motion will begin and end smoothly. Therefore, the constraints  $\theta(t_1) = \phi_1$  and  $\theta(t_L) = \phi_L$  can be considered 'hard' constraints, while the remaining  $(L + 4)$  constraints in (G2.5.11) can be considered 'soft' constraints.

Since the joint trajectory is input to the path tracker at discrete values of time, the trajectory does not need to be a continuous function of time. It can be a discrete set of joint angles, defined only at times  $kh$  ( $k = 0, 1, \dots, N$ ) where  $h$  is the sample period of the path tracker (typically on the order of 0.01 s), and  $Nh$  is the length of the trajectory.

The angle  $\theta_i$  is input to the path tracker every  $h$  s, starting at  $t = 0$  and ending at  $t = T$ . There are exactly  $M$  discrete times per knot, so each knot angle is separated from its neighboring knots by  $Mh$  s. Thus, the path length  $T$  satisfies

$$T = M(L - 1)h. \quad \text{(G2.5.13)}$$

Also, from  $t = 0$  to  $t = T$ , there are exactly  $N + 1$  discrete time steps. Thus, the number of discrete time steps satisfies

$$N + 1 = M(L - 1) + 1. \quad \text{(G2.5.14)}$$

These relationships are depicted graphically in figure G2.5.1.

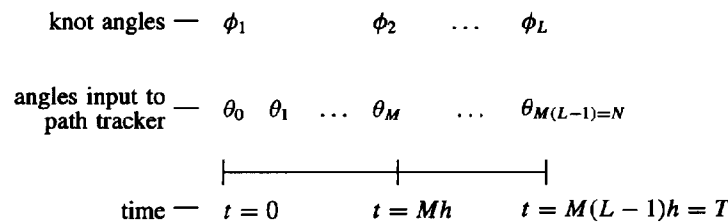


Figure G2.5.1. Relationships between network variables.

So the optimization problem of (G2.5.11) can be discretized (using the the trapezoidal integration rule) into the following problem:

$$\min \left[ \frac{1}{2}(\theta_0''')^2 + \sum_{i=1}^{N-1} (\theta_i''')^2 + \frac{1}{2}(\theta_N''')^2 \right] \quad \text{(G2.5.15)}$$

$$\text{subject to } \begin{aligned} \theta_{M(j-1)} &= \phi_j & (j = 1, \dots, L) \\ \theta'_0 &= 0 \\ \theta'_N &= 0 \\ \theta''_0 &= 0 \\ \theta''_N &= 0 \\ \theta'''_0 &= 0 \\ \theta'''_N &= 0 \end{aligned}$$

where  $\theta_0 = \phi_1$  and  $\theta_{M(L-1)} = \phi_L$  are hard constraints, and the rest of the constraints are soft.

Since the values of  $\theta_0$  and  $\theta_N$  are hard constraints, they can be considered constants. Then the independent variables of the optimization problem are  $\theta_i$  ( $i = 1, \dots, N-1$ ). Note that since we are constraining  $\theta''_0$  and  $\theta''_N$  to zero, they can be omitted from the objective function of (G2.5.15). Then, using finite-difference expressions for the first three derivatives of  $\theta(t)$ , the optimization problem of (G2.5.15) can be converted into the equivalent problem

$$\begin{aligned} \min \sum_{i=1}^{N-1} & (-\theta_{i-2} + 2\theta_{i-1} - 2\theta_{i+1} + \theta_{i+2})^2 & (G2.5.16) \\ \text{subject to } & \theta_{M(j-1)} = \phi_j & (j = 2, \dots, L-1) \\ & \theta_1 = \phi_1 \\ & \theta_2 = \phi_1 \\ & \theta_{N-2} = \phi_L \\ & \theta_{N-1} = \phi_L \end{aligned}$$

where we have defined  $\theta_{-1} \equiv \theta_0$  and  $\theta_{N+1} \equiv \theta_N$ . Now (G2.5.16) can be written as

$$\min(\theta^T \mathbf{A} \theta + \mathbf{b}^T \theta) \text{ subject to } g(\theta) = 0 \quad (G2.5.17)$$

where  $\theta = [\theta_1 \dots \theta_{N-1}]^T$ ,  $g(\theta)$  is the  $(L+2)$ -element constraint vector defined by (G2.5.16), and  $\mathbf{A}$  and  $\mathbf{b}$  are, respectively, an  $(n-1) \times (n-1)$  matrix and an  $(n-1)$ -vector. Matrix  $\mathbf{A}$  is a positive semidefinite matrix of bandwidth 4 (Golub and Van Loan 1989) whose diagonal and first through fourth upper and lower diagonals are given as follows:

$$\begin{aligned} \text{diagonal} &= (5 \ 9 \ 10 \ 10 \ \dots \ 10 \ 10 \ 9 \ 5) \\ \text{first upper and lower diagonal} &= (-2 \ -4 \ -4 \ \dots \ -4 \ -4 \ -2) \\ \text{second upper and lower diagonal} &= (-4 \ -4 \ \dots \ -4 \ -4) \\ \text{third upper and lower diagonal} &= (4 \ 4 \ \dots \ 4 \ 4) \\ \text{fourth upper and lower diagonal} &= (-1 \ -1 \ \dots \ -1 \ -1). \end{aligned} \quad (G2.5.18)$$

Vector  $\mathbf{b}$  is given by

$$\mathbf{b} = (-4\phi_1 \ -4\phi_1 \ 6\phi_1 \ -2\phi_1 \ 0 \ 0 \ \dots \ 0 \ 0 \ -2\phi_L \ 6\phi_L \ -4\phi_L \ -4\phi_L)^T. \quad (G2.5.19)$$

According to the results given by (G2.5.4), (G2.5.17) is solved by the dynamic system

$$\begin{aligned} \dot{\theta} &= -2\mathbf{A}\theta - \mathbf{b} - \frac{\partial g}{\partial \theta}(\lambda + c \circ g) \\ \dot{\lambda} &= c \circ g \end{aligned} \quad (G2.5.20)$$

where  $c \circ g$  is the  $(L+2)$ -vector Hadamard product of  $c$  and  $g$  whose  $i$ th element is given by  $c_i g_i$ . The element in the  $i$ th row and  $j$ th column of  $\partial g / \partial \theta$  is given by  $\partial g_j / \partial \theta_i$ .

If matrix  $\mathbf{A}$  were positive definite, we could set  $c$  equal to the zero vector and still be guaranteed convergence. However, if  $\mathbf{A}$  is only positive semidefinite, we need to use a nonzero  $c$ . Even if  $\mathbf{A}$  is positive definite, a nonzero  $c$  will improve the convergence properties of the neural network.

Note that the neural net may converge to a local minimum rather than a global minimum. Some sort of *simulated annealing* technique can be used in conjunction with the network (Jeffrey and Rosner 1986a, b). This idea results in the long computational time characteristic of annealing, but it also enables the network to find the best solution among many local minima. C1.4.2

The annealing-type method which is suggested is as follows. Once the network converges to a local minimum, the network state is perturbed in a random direction and by a random magnitude. Then the network dynamics are reactivated, and another local minimum is found. During this process, the algorithm keeps track of the best solution. After a predetermined number of local minima are found, the algorithm terminates and the solution with the lowest energy is accepted as the best solution.

### G2.5.3 Comparison with other methods of robot trajectory planning

Two methods were used to generate minimum-jerk robot joint trajectories: a minimum-jerk trigonometric spline method was used by Simon and Isik (1993), and the neural network proposed above was used. The trigonometric spline method is analytical and was coded using MATLAB on a Sun-4 workstation. The neural network is a numerical method and was simulated on a Sun-4 workstation in the C programming language. The neural net dynamics were integrated using a basic fourth-order Runge-Kutta method with an integration step size of 5 ms.

Six multiple-knot, 35-second joint trajectories were calculated using the trigonometric spline method and the simulated neural network. Each joint trajectory has eight evenly spaced knots, corresponding to the examples given in previous work (Lin *et al* 1983, Thompson and Patel 1987).

Plots of the six neural-network-based trajectories which pass through the six sets of knots are given by Simon (1993). The trajectory corresponding to joint 2 (a typical example) is reproduced here in figure G2.5.2. The initial state of the neural nets consisted of the minimum-jerk trigonometric trajectories (Simon and Isik 1993),  $\lambda$  was initialized to the zero vector, and  $c$  was a vector in which each element was 1. Note from figure G2.5.2 that the neural-network-based trajectory does not pass exactly through the knots. The neural network trajectories have small nonzero derivatives at the endpoints. The trigonometric splines have zero velocity, acceleration and jerk at the endpoints, and pass exactly through the knots.

Table G2.5.1 shows the decrease of the jerk objective function due to the evolution of the network dynamics. It is seen that the use of the neural network for this typical example gives an average improvement of almost 20% in the objective function. Although we cannot quantify the result of this decrease at this point in time, we can state that two results are a corresponding decrease in the error of the path tracker, and robot arm movement which appears more smooth and coordinated.

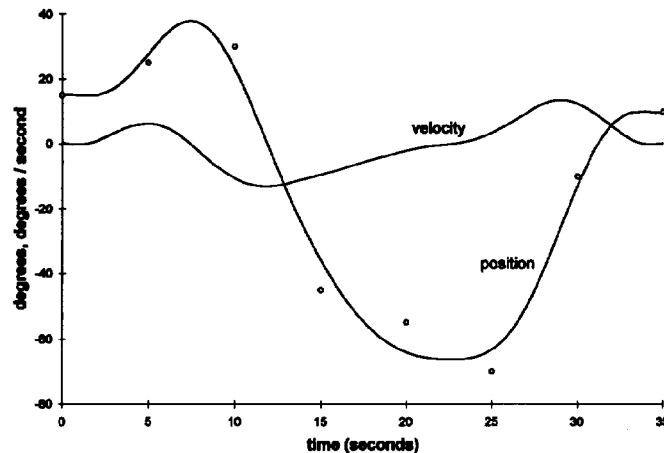


Figure G2.5.2. Minimum-jerk trajectory for joint 2.

### G2.5.4 Conclusions

Minimum-jerk joint trajectories have the properties of similarity to human joint movements (Flanagan and Ostry 1990) and amenability to tracking (Kyriakopoulos and Saridis 1988). This makes them attractive choices for robotics applications in spite of the fact that the dynamics are not taken into account.

In this section, the minimum-jerk joint trajectory formulation problem is posed as a constrained quadratic optimization problem. A hard-wired neural network is proposed to solve the problem numerically.

**Table G2.5.1.** Jerk objective function values.

Joint	Minimum jerk		Decrease (%)
	Trigonometric	Neural net	
1	127	106	16.5
2	44	28	36.3
3	558	462	17.2
4	765	662	13.5
5	252	206	18.3
6	38	33	13.2
Averages	297	250	19.2

The network may converge to a local minimum rather than the global minimum. The solution obtained by the network depends on the initial state of the network. An annealing-type technique is used in conjunction with the network to climb out of local minima and find the best among many solutions. This prevents the algorithm from being appropriate for real-time use, but significantly improves the quality of the final solution. The simulation results presented verify that the network can be successfully applied to robot trajectory generation.

The neural-network-generated trajectories pass near but not exactly through the specified knots. If it is important that the trajectory pass exactly through the knots, this method may not be suitable for joint interpolation. While this section has dealt specifically with minimum-jerk joint trajectories, there are no theoretical limitations to applying this method to other objective functions. More specifically, minimum-energy or minimum-torque-change trajectories could be generated with the network discussed in this section.

### Acknowledgements

Much of this section has been adapted from the article by Simon (1993) where additional details can be found, and the permission of the publisher is gratefully acknowledged.

### References

- Arrow K, Hurwicz L and Uzawa H 1958 *Studies in Linear and Nonlinear Programming* (Stanford, CA: Stanford University Press)
- Cohen M and Grossberg S Absolute stability of global pattern formation and parallel memory storage by competitive neural networks *IEEE Trans. Systems, Man, Cybern.* **13** 815–26
- Craig J 1989 *Introduction to Robotics* (Reading, MA: Addison-Wesley)
- Flanagan J and Ostry D 1990 Trajectories of human multi-joint arm movements: evidence of joint level planning *Experimental Robotics I, 1st Int. Symp.* ed V Hayward and O Khatib (New York: Springer)
- Flash T and Hogan N 1985 The coordination of arm movements: an experimentally confirmed mathematical model *J. Neurosci.* **5** 1688–703
- Golub G and Van Loan C 1989 *Matrix Computations* 2nd edn (Baltimore, MD: Johns Hopkins University Press)
- Jang J *et al* 1988 An optimization network for matrix inversion *Neural Information Processing Systems* ed D Anderson (New York: American Institute of Physics) pp 397–401
- Jeffrey W and Rosner R 1986a Optimization algorithms: simulated annealing and neural network processing *Astrophys. J.* **310** 473–81
- 1986b Neural network processing as a tool for function optimization *Neural Networks for Computing* ed J Denker (New York: American Institute of Physics) pp 241–6
- Kawato M *et al* 1990 Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion *Biol. Cybern.* **62** 275–88
- Kyriakopoulos K and Saridis G 1988 Minimum jerk path generation *IEEE Int. Conf. on Robotics and Automation* vol 1, pp 364–9
- Lin C and Chang P 1983 Joint trajectories of mechanical manipulators for Cartesian path approximation *IEEE Trans. Syst. Man Cybern.* **13** 1094–102
- Lin C, Chang P and Luh J 1983 Formulation and optimization of cubic polynomial joint trajectories for industrial robots *IEEE Trans. Automatic Control* **28** 1066–73

- Platt J and Barr A 1988 Constrained differential optimization *Neural Information Processing Systems* ed D Anderson (New York: American Institute of Physics) pp 612-21
- Shih L 1984 On the elliptic path of an end-effector for an anthropomorphic manipulator *Int. J. Robot. Res.* **3** 51-7
- Shiller Z and Dubowsky S 1989 Robot path planning with obstacles, actuator, gripper, and payload constraints *Int. J. Robot. Res.* **8** 3-18
- Simon D 1993 The application of neural networks to optimal robot trajectory planning *Robot. Autonomous Syst.* **11** 23-34
- Simon D and Isik C 1993 A trigonometric trajectory generator for robotic arms *Int. J. Control* **57** 505-17
- Thompson S and Patel R 1987 Formulation of joint trajectories for industrial robots using B-splines *IEEE Trans. Indust. Electron.* **34** 192-9
- Uno Y *et al* 1989 Formation and control of optimal trajectory in human multijoint arm movement *Biol. Cybern.* **61** 89-101
- Zhao X and Mendel J 1988 An artificial neural minimum-variance estimator *IEEE Conf. on Neural Networks* vol 2, pp 499-506