Cleveland State University
EngagedScholarship@CSU

Electrical Engineering & Computer Science Faculty Publications

Electrical Engineering & Computer Science Department

The Application of Neural Networks to Optimal Robot Trajectory Planning

Daniel J. Simon
Cleveland State University, d.j.simon@csuohio.edu

Follow this and additional works at: https://engagedscholarship.csuohio.edu/enece_facpub

Part of the Robotics Commons

How does access to this work benefit you? Let us know!

Publisher's Statement

NOTICE: this is the author's version of a work that was accepted for publication in Robotics and Autonomous Systems. Changes resulting from the publishing process, such as peer review, editing, corrections, structural formatting, and other quality control mechanisms may not be reflected in this document. Changes may have been made to this work since it was submitted for publication. A definitive version was subsequently published in Robotics and Autonomous Systems, 11, 1, (05-01-1993); dx.doi.org/10.1016/0921-8890(93)90005-W

Original Citation

D. Simon. (1993). The Application of Neural Networks to Optimal Robot Trajectory Planning. Robotics and Autonomous, 11(1), pp. 23-34.

Repository Citation

Simon, Daniel J., "The Application of Neural Networks to Optimal Robot Trajectory Planning" (1993). *Electrical Engineering & Computer Science Faculty Publications*. 130.
https://engagedscholarship.csuohio.edu/enece_facpub/130

The application of neural networks to optimal robot trajectory planning

Dan Simon

TRW Ballistic Missiles Division, Space and Technology Group, P.O. Box 1310, Building SB2, Room 1062, San Bernardino, CA 92402-1310, USA

Abstract

Interpolation of minimum jerk robot joint trajectories through an arbitrary number of knots is realized using a hardwired 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 rather than analytic functions of time. This application formulates the interpolation problem 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 neuron outputs define the joint angles (one neuron for each discrete value of time) and the Lagrange multipliers (one neuron for each trajectory constraint). An annealing-type method is used to prevent the network from getting stuck in a local minimum. This paper discusses the optimizing neural network and its application to robot path planning, presents some simulation results, and compares the neural network method with other minimum jerk trajectory planning methods.

Keywords: Neural network: Optimization; Robot path planning: Minimum jerk.

1. Introduction

This section first gives a brief review of robot path planning. Then an introduction to robot path planning using neural networks is presented.

1.1. Path 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* [4]. 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. The details of the robot dynamics and constraints are taken into account at various levels of complexity. A concise review of such algorithms is given in [30]. While such methods are attractive in that they provide optimal solutions to some robot control problem, they result in impractically complicated algorithms and a large computational expense. For this reason, several researchers [11,32,34,37] have simplified the problem as follows: given a desired path in Cartesian space (e.g. a straight line), derive the optimum joint trajectories subject to the full dynamics and constraints. This approach assumes that the Cartesian path is already specified. While this assumption reduces the computational expense somewhat, the expense is still much too high for on-line implementation.

Some researchers [9,31,33] do not assume that the path is specified, but that it has some known form (e.g. algebraic or trigonometric spline). They then optimize the spline parameters with respect to some objective function.

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 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 analytic interpolating curve provides the path tracker with joint angles and derivatives at the controller rate.

The most popular type of interpolation is algebraic splines [21,22,35]. Higher-order splines result in continuity of higher-order derivatives, which reduces wear and tear on the robot [4], but this is at the expense of large oscillations of the

trajectory. Trigonometric splines can be used to provide a less oscillatory interpolating curve [33].

1.2. Neural networks applied to robot path planning

Neural networks have been applied to many fields of engineering, and the field of robotics is no exception [38]. Robotics applications of neural nets include object recognition [20], dynamics identification [19], control [29], path planning [16], inverse kinematics [10], trajectory generation [26], and task scheduling [23]. This paper presents a new application of a hardwired optimization network to the problem of trajectory generation through a set of given knots. The optimization network is in the form of first-order differential equations, and can thus be directly implemented in hardware. Performing constrained optimization at the raw speed of VLSI seems like a promising technique for solving large-scale robot trajectory planning problems.

The robot path-planning problem can be viewed as a constrained 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'. It is shown in this paper that the robot path-planning problem can be reduced to a constrained quadratic programming problem. Closed form quadratic programming methods cannot be used to solve the problem due to the fact that the problem is not positive definite, and the constraints are not full rank. Classical iterative quadratic programming methods (such as Hildreth's method [24] or Newton's method) could be used, but typically require a good deal of computation at each iteration [25].

The past decade has seen a proliferation of more 'natural' methods of optimization for large-scale problems. Only a small fraction can be mentioned here. For instance, simulated annealing mimics the freezing of liquids or the annealing of metals. It has solved difficult problems, but the required computational times are often excessive. Some recent work has focused on speeding up simulated annealing methods [15,27]. Hopfield-type nets have also been used on large-scale optimization problems. Many of these networks have restrictions (such as limits on the norm of the quadratic weight matrix) which limit their applications [13,14,39]. In [12] a neural network

for finding the inverse or pseudoinverse of a matrix is given. Such a network could be used for quadratic optimization. Kohonen nets, or Learning Vector Quantizers, have also been applied to combinatorial optimization problems [1,7]. Platt and Barr [28] formulate a neural network which can calculate a minimum of a general function subject to inequality or equality constraints. This is the network which is used in this paper to determine a minimum jerk robot joint path through a given set of knots. The reason that Platt and Barr's network is used in this paper is that it can be shown to be locally stable for quadratic problems. But in view of the many different network architectures which have been used for optimization, some other network may indeed be more suitable for robot path planning. The purpose of this paper is not to find the 'best' path planning network, but rather to demonstrate the applicability of neural nets to this problem.

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. Previous work [5,6] suggests that human arm movements minimize a measure of Cartesian jerk or joint jerk. Others [17,36] 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 paper, a joint jerk objective function is used. While this choice ignores the dynamics of the robot, it reduces the error of the path tracker [18] and thus is suitable for robotics applications. Pontryagin's minimum principle has been used to analytically determine minimum joint jerk trajectories between two points subject to the constraints of zero velocity and acceleration at the endpoints [18]. The minimum Cartesian jerk trajectory between two points has also been derived [36]. But numerical methods must be used if more than two knots are given. This paper presents a method which is used to plan minimum joint jerk trajectories through an arbitrary number of knots.

Section 2 of this paper discusses the neural network architecture which is used, and Section 3 applies the network to the robot trajectory formulation problem. Section 4 presents some simulation results, and Section 5 presents some concluding remarks.

2. Constrained minimization using neural networks

Platt and Barr [28] formulate a neural network which can be used for constrained minimization. Their algorithm, along with some straightforward extensions, is summarized in this section.

Consider the following constrained minimization problem:

$$\min f(\vec{x}) \text{ subject to } \vec{g}(\vec{x}) = 0. \tag{1}$$

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

Lagrange multipliers can be used to convert the constrained problem of Eq. (1) to the following unconstrained problem:

$$\min\left[f(\vec{x}) + \vec{\lambda}^T \vec{g}(\vec{x}) \right], \tag{2}$$

where $\vec{\lambda}$ is an m-vector of Lagrange multipliers associated with the constraints $\vec{g}(\cdot)$. A necessary condition for the solution of Eq. (2) is

$$\frac{\partial f}{\partial \vec{x}} + \vec{\lambda}^T \frac{\partial \vec{g}}{\partial \vec{x}} = 0. \tag{3}$$

Now consider a neural network with dynamics of the form

$$\dot{x}_i = -\frac{\partial f}{\partial x_i} - \sum_{\alpha=1}^{m} (\lambda_\alpha + c_\alpha g_\alpha) \frac{\partial g_\alpha}{\partial x_i}$$

$$(i = 1, \ldots, n),$$

$$\dot{\lambda}_j = c_j g_j \qquad (j = 1, \ldots, m). \tag{4}$$

where c is an m-vector of constants. Assume that the constraints $\vec{g}(\cdot)$ of the original problem (Eq. (1)) are linear functions of x. Then differentiating $\dot{x}_i$ in Eq. (4) gives

$$\ddot{x}_i + \sum_{j=1}^{n} \frac{\partial^2 f}{\partial x_i \partial x_j} \dot{x}_j$$

$$+ \sum_{\alpha=1}^{m} \left[c_\alpha \left(g_\alpha + \sum_{j=1}^{n} \frac{\partial g_\alpha}{\partial x_j} \dot{x}_j \right) \frac{\partial g_\alpha}{\partial x} \right] = 0. \tag{5}$$

Now consider the candidate Lyapunov energy function

$$E = \tfrac{1}{2} \sum_{i=1}^{n} (\dot{x}_i)^2 + \tfrac{1}{2} \sum_{\alpha=1}^{m} c_\alpha g_\alpha^2. \tag{6}$$

The derivative of this energy function is

$$\dot{E} = \sum_{i=1}^{n} \left(\dot{x}_i \ddot{x}_i + \sum_{\alpha=1}^{m} c_\alpha g_\alpha \frac{\partial g_\alpha}{\partial x_i} \dot{x}_i \right) \tag{7}$$

$$= - \sum_{i,j=1}^{n} \dot{x}_i \dot{x}_j \left(\frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{\alpha=1}^{m} c_\alpha \frac{\partial g_\alpha}{\partial x_j} \frac{\partial g_\alpha}{\partial x_i} \right) \tag{8}$$

$$= -\dot{\vec{x}}^T A \dot{\vec{x}}, \tag{9}$$

where the element in the ith row and jth column of matrix A is given by

$$A_{i,j} = \frac{\partial^2 f}{\partial x_i \partial x_j} + \sum_{\alpha=1}^{m} c_\alpha \frac{\partial g_\alpha}{\partial x_j} \frac{\partial g_\alpha}{\partial x_i}. \tag{10}$$

It has been shown [2,3,28] that there exists a finite vector $\vec{c}$ such that matrix A is positive definite at the constrained minima of Eq. (1). If A is continuous, then it is positive definite in some region surrounding each constrained minimum. Therefore if the dynamic system defined by Eq. (4). begins in that region and remains in that region, $\dot{E}$ will remain less than zero unless $\dot{x}_i = 0 \forall$ i. The energy E will therefore achieve a minimum when $\dot{x}_i = 0 \forall$ i. But E is not a minimum unless each of the constraints $\vec{g}(\vec{x})$ are zero, since $\vec{g}(\cdot)$ is linear. Therefore the system will settle into the zero-energy state where

$$\dot{\vec{x}} = 0. \tag{11}$$

$$\vec{g}(\vec{x}) = 0. \tag{12}$$

Now $\vec{g}(\vec{x}) = 0$ implies that the original constraints are satisfied, and $\dot{\vec{x}} = 0$ implies (Eq. (4)) that

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

$$= \frac{\partial f}{\partial x_i} + \sum_{\alpha=1}^{m} \lambda_\alpha \frac{\partial g_\alpha}{\partial x_i} \tag{13}$$

$$= \frac{\partial f}{\partial \vec{x}} + \vec{\lambda}^T \frac{\partial \vec{g}}{\partial \vec{x}}$$

$$= 0,$$

which satisfies the necessary conditions for a local minimum of the original constrained problem (see Eq. (3)).

To sum up, Eq. (4), with an appropriately chosen $\vec{c}$, converges to a solution of the original constrained minimization problem of Eq. (1). Equation (4) is in the form of a first-order differ-

ential equation, which implies that it can be implemented in parallel hardware to yield a very quick solution.

3. The solution of the robot path planning problem

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 [5], reduces wear on the robot joints and prevents the excitation of resonances [4], and improves the accuracy of the path tracker [18]. 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; and
- has 'small' derivatives at the endpoints.
Smoothness can be defined as the integral of the square of the jerk of the position trajectory [5]. 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 seconds, and the desired knot angles are $\theta(t_j) = \theta_j (j = 1, \ldots, L)$, then the optimization problem for each joint can be written as

$$\min \int_0^T [\theta'''(t)]^2 \, dt \quad \text{subject to} \tag{14}$$

$$\theta(t_j) = \phi_j \quad (j = 1, \ldots, L)$$

$$\theta'(0) = 0$$

$$\theta'(T) = 0$$

$$\theta''(0) = 0$$

$$\theta''(T) = 0$$

$$\theta'''(0) = 0$$

$$\theta'''(T) = 0.$$

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

$$t_i = (i - 1)T/(L - 1) \quad (i = 1, \ldots, L). \tag{15}$$

The joint trajectory at the endpoint 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 Eq. (14) 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 $(j = 0, 1, \ldots, N)$ where h is the sample period of the path tracker (typically on the order of 0.01 seconds), and Nh is the length of the trajectory.

The angle θ_j is input to the path tracker every h seconds, 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 seconds. Thus the path length T satisfies.

$$T = M(L - 1)h. \tag{16}$$

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

$$N + 1 = M(L - 1) + 1. \tag{17}$$

These relationships are depicted graphically in *Fig. 1*. So the optimization problem of Eq. (14) can be discretized (with the help of the trapezoidal integration rule) into the following problem.

$$\min \left[\tfrac{1}{2}(\theta_0''')^2 + \sum_{i=1}^{N-1} (\theta_i''')^2 + \tfrac{1}{2}(\theta_N''')^2 \right] \quad \text{subject to} \tag{18}$$

$$\theta_{M(j-1)} = \phi_j \quad (j = 1, \ldots, L)$$

$$\theta_0' = 0$$
$$\theta_N' = 0$$
$$\theta_0'' = 0$$
$$\theta_N'' = 0$$
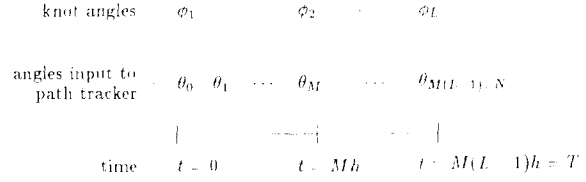$$\theta_0''' = 0$$
$$\theta_N''' = 0,$$

Fig. 1. Relationships between network variables.

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

Finite difference expressions for the first three derivatives of $\theta(t)$ can be formed using Taylor series. These finite differences are as follows:

$$\theta'''(t) = \left[-\theta(t - 2h) + 2\theta(t - h) \right.$$
$$\left. -2\theta(t + h) + \theta(t + 2h) \right] / (2h^3)$$
$$+ O(h^2).$$

$$\theta''(t) = \left[\theta(t + h) - 2\theta(t) + \theta(t - h) \right] / h^2$$
$$+ O(h^2),$$

$$\theta'(t) = \left[\theta(t + h) - \theta(t - h) \right] / (2h) + O(h^2). \tag{19}$$

Since the values of θ_0 and θ_N are hard constraints, they can be considered constants. Then the independent variables of the optimization problem are $\theta_i(i = 1, \ldots, N - 1)$. Note that since we are constraining θ_0''' and θ_N''' to zero, they can be omitted from the objective function of Eq. (18). Then, using Eq. (19), the optimization problem of Eq. (18) can be converted into the equivalent problem

$$\min \sum_{i=1}^{N-1} (-\theta_{i-2} + 2\theta_{i-1} - 2\theta_{i+1} + \theta_{i+2})^2$$

subject to $\qquad (20)$

$$\theta_{M(j-1)} = \phi_j \quad (j = 2, \ldots, L - 1)$$

$$\theta_1 = \phi_1$$
$$\theta_2 = \phi_1$$
$$\theta_{N-2} = \phi_L$$
$$\theta_{N-1} = \phi_L,$$

where we have defined $\theta_{-1} \equiv \theta_0$ and $\theta_{N+1} \equiv \theta_N$. Now Eqs. (20) can be written as

$$\min\left(\vec{\theta}^T A \vec{\theta} + \vec{b}^T \vec{\theta}\right) \quad \text{subject to} \quad \vec{g}(\vec{\theta}) = 0, \tag{21}$$

where $\vec{\theta} = [\theta_1 \cdots \theta_{N-1}]^T$, $g(\vec{\theta})$ is the $(L + 2)$-element constraint vector defined by Eq. (20),

and A and $\vec{b}$ are respectively an $(n-1) \times (n-1)$ matrix and an $(n-1)$-vector. Matrix A is a positive semidefinite matrix of bandwidth four whose diagonal and first through fourth upper and lower diagonals are given as follows. (See [8] for a definition of 'matrix bandwidth').

diagonal $= (5\ 9\ 10\ 10\ \cdots\ 10\ 10\ 9\ 5)$

first upper and lower diagonal

$$= (-2\ -4\ -4\ \cdots\ -4\ -4\ -2)$$

second upper and lower diagonal

$$= (-4\ -4\ \cdots\ -4\ -4)$$

third upper and lower diagonal

$$= (4\ 4\ \cdots\ 4\ 4)$$

fourth upper and lower diagonal

$$= (-1\ -1\ \cdots\ -1\ -1). \tag{22}$$

Vector $\vec{b}$ is given by

$$\vec{b} = (-4\phi_1\ -4\phi_1\ 6\phi_1\ -2\phi_1\ 0\ \cdots\ 0\ 0$$

$$-2\phi_L\ 6\phi_L\ -4\phi_L\ -4\phi_L)^{\mathrm{T}}. \tag{23}$$

According to the results given by Eq. (4), Eq. (21) is solved by the dynamic system

$$\dot{\vec{\theta}} = -2A\vec{\theta} - \vec{b} - \frac{\partial \vec{g}}{\partial \vec{\theta}}\left(\vec{\lambda} + \vec{c} \circ \vec{g}\right),$$

$$\vec{\lambda} = \vec{c} \circ \vec{g}, \tag{24}$$

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

Since matrix A has bandwidth four, the neural network is not fully connected. In fact, the interconnections are quite sparse, which reduces the complexity of the network. Each θ neuron receives inputs only from itself and its four closest neighbors on each side, and from one of the λ neurons. Each λ neuron receives inputs only from one of the θ neurons.

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

It is important to note that the neural net considered in this paper may converge to a local minimum rather than a global minimum. A given trajectory planning problem, discretized into $(N+1)$ joint angles, is a function of $(N-1)$ variables (see Eq. (20)) and may have many local minima. The solution to which the neural net converges depends on the initial state of the network. Some sort of simulated annealing technique can be used in conjunction with the network described in this paper [13,14]. This idea results in the longer computational time characteristic of annealing, but it also enables the network to find the best solution among many local minima.

The annealing-type method which is suggested in this paper 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.

This method is thus a cross between an exhaustive search and simulated annealing. It is not an exhaustive search, because a relatively few number of initial states are chosen from which the network converges to a minimum. But it is not stimulated annealing either, because the network energy (in this case the integral of the square of the jerk) always decreases and there is no cooling schedule.

4. Simulation results

The neural network proposed in the previous section was simulated on a Sun-4 Workstation in the C language. The neural net dynamics were integrated using a basic fourth-order Runge-Kutta method with an integration step size of 5 msec.

Six multiple-knot joint trajectories were calculated using the simulated neural network. Each joint trajectory has eight evenly spaced knots, corresponding to the examples given in previous work [22,33,35]. The knots were chosen along a desired path of the end-effector of a Unimate PUMA 560 type robot with six revolute joints. Each path length is 35 seconds. The joint space knot angles are given in *Table 1*. Each trajectory
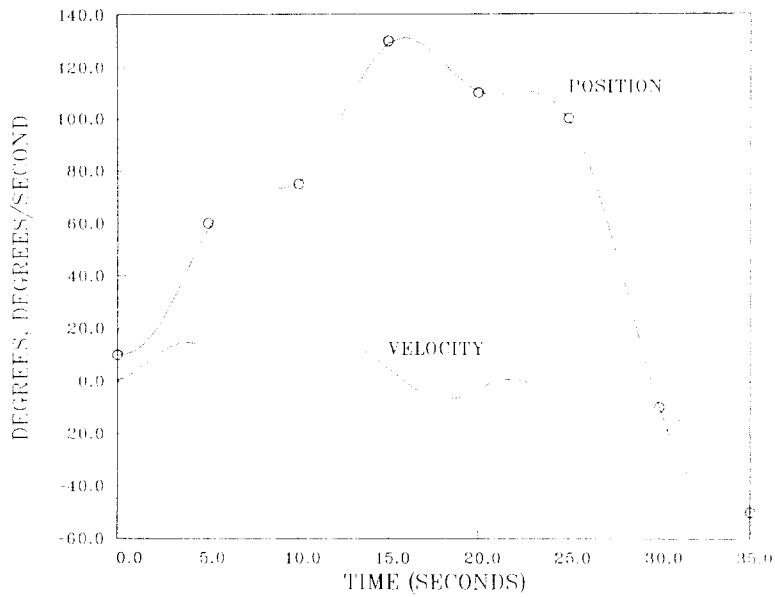
Fig. 2. Joint 1 minimum jerk trajectory.

was discretized into 71 time points. So the neural network had 81 neurons, one for each time point and one for each trajectory constraint (see Eq. (20)). The parameters shown in *Fig. 1* were $L = 8$, $M = 10$, $N = 70$, $h = 0.5$ seconds, and $T = 35$ seconds. A real application would require much finer discretization, perhaps on the order of $h = 0.01$ seconds (the controller rate). The results
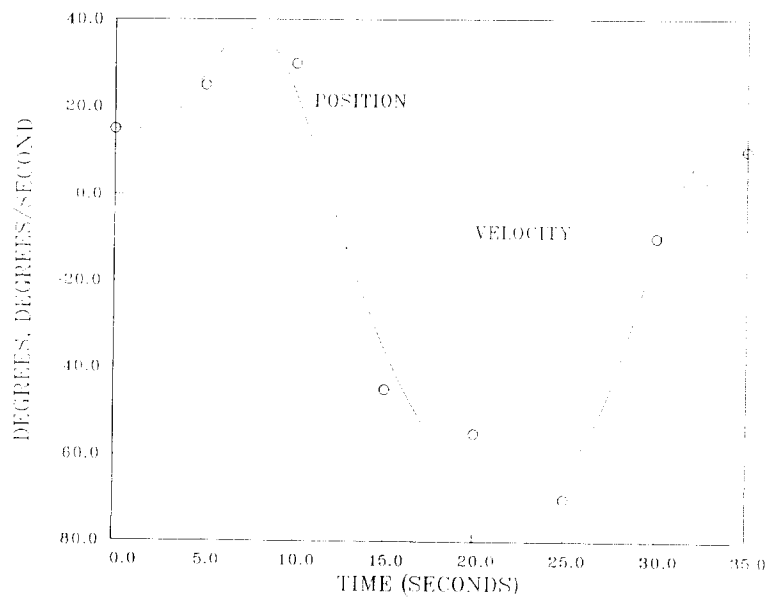

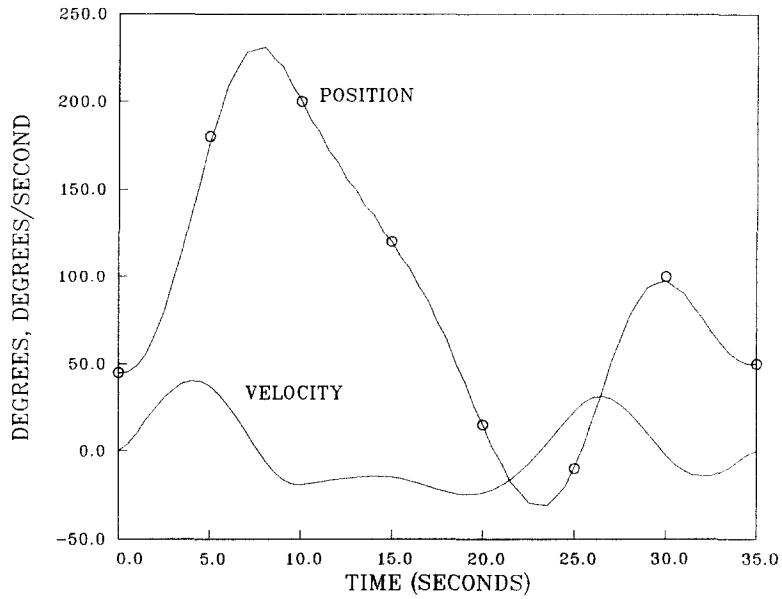Fig. 3. Joint 2 minimum jerk trajectory.

Fig. 4. Joint 3 minimum jerk trajectory.

given in this section are only intended to demonstrate the feasibility of the proposed trajectory planning method. If $h = 0.5$ was used for a real robot, the neural network solution could be interpolated by some standard method to give joint angles at the controller rate.

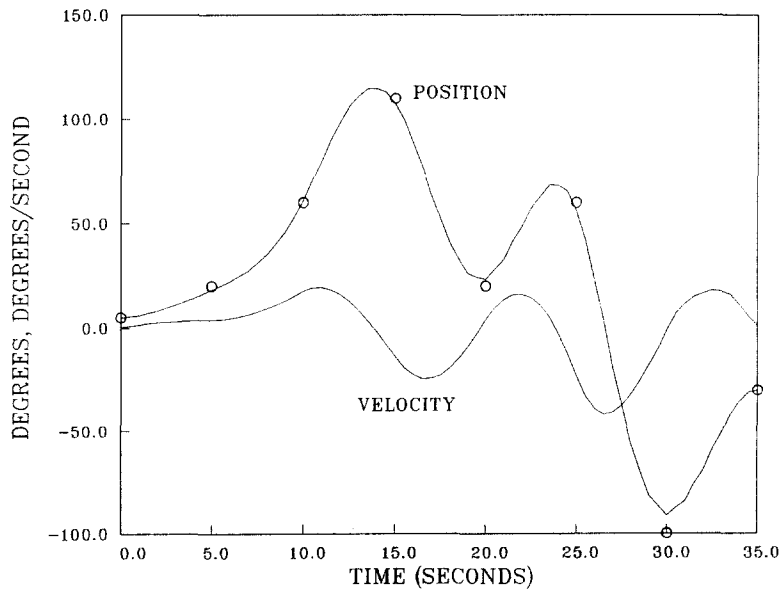Plots of the six neural network based trajectories which pass through the six sets of knots are

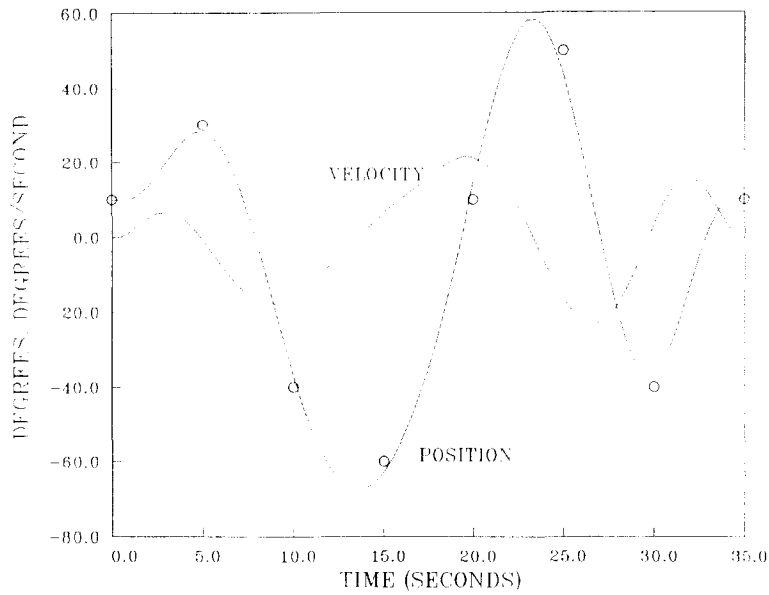Fig. 5. Joint 4 minimum jerk trajectory.

Fig. 6. Joint 5 minimum jerk trajectory.

given in *Figs. 2–7*. The initial state of the neural nets consisted of minimum jerk trigonometric trajectories [33], $\vec{\lambda}$ was initialized to the zero vector, and $\vec{c}$ was a vector in which each element was 1.

Note from the figures that the neural network based trajectory does not exactly pass through the knots. If the user desired the trajectory to pass closer to the knots, the relative weights of the
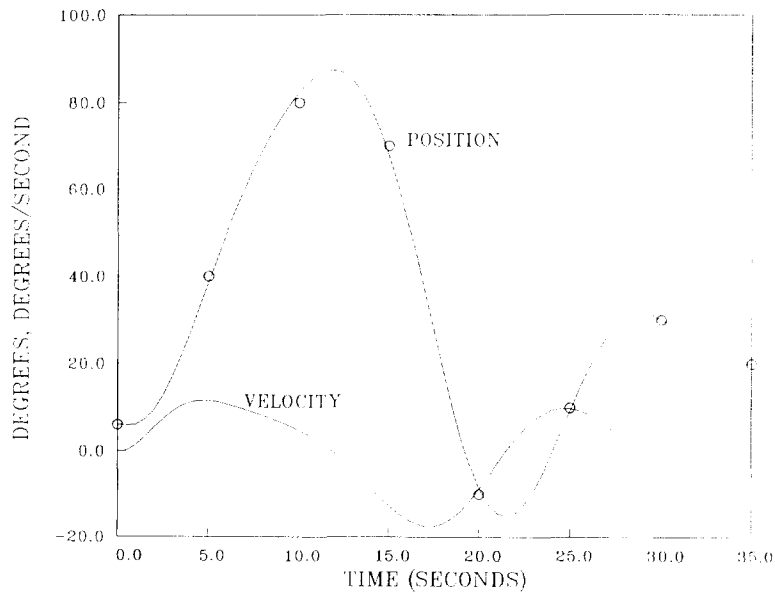

Fig. 7. Joint 6 minimum jerk trajectory.

Table 1
PUMA 560 knot angles

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 3
Effect of constraint weight $\vec{c}$ on joint 1 trajectory

Constraint weight $\vec{c}$	Objective function value	RMS knot angle error (degrees)	Max knot angle error (degrees)
0.2	87	6.5	9.6
0.4	93	6.4	9.4
0.6	96	5.6	8.3
0.8	99	5.3	7.8
1.0	101	5.0	7.3
1.2	103	4.6	6.7
1.4	105	4.1	6.0

knot constraints can be increased (see Eqs. (20)–(21)). *Table 2* shows the decrease of the jerk objective function due to the evolution of the network dynamics. It should be noted that the trigonometric splines have zero velocity, acceleration, and jerk at the endpoints, and pass exactly through the knots. The neural network trajectories have a small nonzero velocity, acceleration, and jerk at the endpoints. In addition, they pass near but not exactly through the knots.

Due to the nonlinearity of the network, changes in network parameters lead to unpredictable results. The constraint weights $\vec{c}$, the initial state of the network, and the annealing schedule all have a hand in determining the local minimum to which the network eventually converges. In general, it is expected that an increase in constraint weights $\vec{c}$ leads to a decrease in the constraint violations and an increase in the objective function value. This is shown in *Table 3* for joint 1. (Note that the results in *Table 3* do not correspond to *Fig. 2* or *Table 2* because different

initial conditions and annealing strategies were used.)

5. Conclusion

Minimum jerk joint trajectories have the properties of similarity to human joint movements [5] and amenability to tracking [18]. This makes them attractive choices for robotics applications in spite of the fact that the dynamics are not taken into account. Analytic formulations of minimum jerk trajectories between two points are known [18,36]. But if there are more than two knots, analytic solutions cannot be obtained and numerical solutions must be used.

In this paper, the minimum jerk point trajectory formulation problem is posed as a constrained quadratic optimization problem. The joint angle domain is continuous, and the time domain is discretized at the robot controller rate. Several neural network architectures have been proposed for quadratic optimization over a continuous domain. Most of those proposed, however, impose restrictions on the quadratic weighting matrix in order for convergence to be assured. The network used in this paper does not impose any restrictions on the quadratic weighting matrix. The minimum jerk problem is thus amenable to implementation by a neural network.

The network discussed in this paper 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 minimum and find the best among many solutions. This prevents the algorithm from being appropriate

Table 2
Jerk objective function values

Joint	Minimum jerk trigonometric	Minimum jerk neural net	Percent decrease
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

for real-time use, but significantly improves the quality of the final solution. The simulation results presented in this paper verify that the network can be successfully applied to robot trajectory generation.

Note that 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 paper 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 paper.

References

[1] B. Angeniol, G. De La Croix Van Brois and J.-Y. Le Texier, Self-organizing maps and the traveling salesman problem, *Neural Networks* 1 (1988) 289–293.

[2] K. Arrow, L. Hurwicz and H. Uzawa, *Studies in Linear and Nonlinear Programming* (Stanford University Press, Stanford, CA, 1958).

[3] D. Bertsekas, *Automatica* 12 (1976) 133–145.

[4] J. Craig, *Introduction to Robotics* (Addison-Wesley, Reading, MA, 1989).

[5] J. Flanagan and D. Ostry, Trajectories of human multi-joint arm movements: Evidence of joint level planning, in: V. Hayward and O. Khatib, eds., *Experimental Robotics I, The First International Symposium* (Springer-Verlag, New York, 1990).

[6] T. Flash and N. Hogan, The coordination of arm movements: an experimentally confirmed mathematical model, *Journal of Neuroscience* 5 (1985) 1688–1703.

[7] B. Fritzke and P. Wilke, FLEXMAP – A neural network for the travelling salesman problem with linear time and space complexity, *International Joint Conference on Neural Networks* 2 (1991) 929–934.

[8] G. Golub and C. van Loan, *Matrix Computation* (2nd ed.) (The Johns Hopkins University Press, Baltimore, MD, 1989).

[9] J. Bobrow, Optimal robot path planning using the minimum-time criterion, *IEEE Journal of Robotics and Automation* 4 (1988) 443–449.

[10] J. Guo and V. Cherkassky, A solution to the inverse kinematics problem in robotics using neural network processing, *International Joint Conference on Neural Networks* 2 (1989) 299–304.

[11] A. Hejase, Optimal robot trajectory planning using dynamic models, Ph.D. Dissertation, Department of Electrical Engineering, Syracuse University, Syracuse, NY, 1987.

[12] J. Jang et al., An optimization network for matrix inversion, in: D. Anderson, ed., *Neural Information Processing Systems* (American Institute of Physics, New York, 1988) 397–401.

[13] W. Jeffrey and R. Rosner, Optimization Algorithms: Simulated annealing and neural network processing, *The Astrophysical Journal* 310 (1) part 1 (1986) 473–481.

[14] W. Jeffrey and R. Rosner, Neural network processing as a tool for function optimization, in: J. Denker, ed., *Neural Networks for Computing* (American Institute of Physics, New York, 1986) 241–246.

[15] H. Jeong and J. Park, Lower bounds of annealing schedule for Boltzmann and Cauchy machines, *IEEE INNS International Conference on Neural Networks* 1 (1989) 581–586.

[16] C. Jorgensen, Neural network representation of sensor graphs in autonomous robot path planning, *First IEEE International Conference on Neural Networks* 4 (1987) 507–515.

[17] M. Kawato, Y. Uno and R. Suziki, Trajectory formation of arm movement by cascade neural network model based on minimum torque-change criterion, *Biological Cybernetics* 62 (1990) 275–288.

[18] K. Kyriakopoulos and G. Saridis, Minimum jerk path generation, *IEEE International Conference on Robotics and Automation* 1 (1988) 364–369.

[19] M. Leahy et al., Neural network payload estimation for adaptive robot control, *IEEE Transactions on Neural Networks* 2 (1991) 93–100.

[20] S. Lee et al., A computer vision architecture for intelligent control of neural recognition in dynamic environments, *AAAI Workshop on Neural Architectures for Computer Vision* (1988).

[21] C. Lin and P. Chang, Joint trajectories of mechanical manipulators for Cartesian path approximation, *IEEE Transactions on Systems, Man, and Cybernetics* SMC-13 (1983) 1094–1102.

[22] C. Lin, P. Chang and J. Luh, Formulation and optimization of cubic polynomial joint trajectories for industrial robots, *IEEE Transactions on Automatic Control* AC-28 (1983) 1066–1073.

[23] Z. Lo and B. Bavarian, A modified Hopfield neural network for optimization of manufacturing task scheduling, in: M. Jamshidi and M. Saif, eds., *Robotics and Manufacturing: Recent Trends in Research, Education, and Applications* (ASME Press, New York, 1990) vol. 3, 1001–1006.

[24] D. Luenberger, *Optimization by Vector Space Methods* (Wiley, New York, 1968).

[25] D. Luenberger, *Linear and Nonlinear Programming* (Addison-Wesley, Reading, MA, 1984).

[26] L. Massone and E. Bizzi, Generation of limb trajectories with a sequential network, *International Joint Conference on Neural Networks* 2 (1989) 345–349.

[27] I. Matsuba, Optimal simulated annealing method and its application to combinatorial problems, *IEEE INNS International Joint Conference on Neural Networks* 1 (1989) 541–546.

[28] J. Platt and A. Barr, Constrained differential optimization, in: D. Anderson, ed., *Neural Information Processing Systems* (American Institute of Physics, New York, 1988) 612–621.

[29] D. Psaltis et al., Neural controllers, *First IEEE International Conference on Neural Networks* 4 (1987) 551–558.

[30] Z. Shiller and S. Dubowsky, Robot path planning with obstacles, actuator, gripper, and payload constraints, *The International Journal of Robotics Research* 8 (1989) 3–18.

[31] Z. Shiller and H. Lu, Robust computation of path constrained time optimal motions, *IEEE International Conference on Robotics and Automation* 1 (1990) 144–149.

[32] K. Shin and N. McKay, Minimum-time control of robotic manipulators with geometric path constraints, *IEEE Transactions on Automatic Control* AC-30 (1985) 531–541.

[33] D. Simon and C. Isik, Optimal trigonometric joint trajectories, *Robotica* 9 (1991) 379–386.

[34] H. Tan and R. Potts, A discrete trajectory planner for robotic arms with six degrees of freedom, *IEEE Transactions on Robotics and Automation* 5 (1989) 681–690.

[35] S. Thompson and R. Patel, Formulation of joint trajectories for industrial robots using B-splines, *IEEE Transactions on Industrial Electronics* IE-34 (1987) 192–199.

[36] Y. Uno, M. Kawato and R. Suziki, Formation and control of optimal trajectory in human multijoint arm movement, *Biological Cybernetics* 61 (1989) 89–101.

[37] M. Vukobratović and M. Kirćnski, *Scientific Fundamentals of Robotics 3: Kinematics and Trajectory Synthesis of Manipulator Robots* (Springer-Verlag, New York, 1986).

[38] P. Wasserman and R. Oetzel, *Neural Source: The Bibliographic Guide to Artificial Neural Networks*, (Van Nostrand Reinhold, New York, 1990) 960–966.

[39] X. Zhao and J. Mendel, An artificial neural minimum-variance estimator, *IEEE Conference on Neural Networks* 2 (1988) 499–506.