Neuro-adaptive hybrid position/force control of robotic manipulators

S.M. Ziauddin and A.M.S. Zalzala

Robotics Research Group

Department of Automatic Control and Systems Engineering

University of Sheffield

Email: rrg@sheffield.ac.uk

Research Report no. 543

6 October 1994

Abstract

This paper presents a neural network approach to hybrid control of manipulators interacting with the environment. The overall control strategy comprises a nominal model of the manipulator with separate neural network compensators along the force and motion controlled directions in the task co-ordinate system. With the learning mechanism in the task space, modelling errors, dynamic friction and changes in environment stiffness are automatically compensated for which result in highly desirable task oriented performance characteristics. Simulation results are provided using the PUMA 560 arm which demonstrates the applicability of the proposed method to the position / force hybrid control of manipulators.

Neuro-adaptive hybrid position/force control of robotic manipulators

Abstract

This paper presents a neural network approach to hybrid control of manipulators interacting with the environment. The overall control strategy comprises a nominal model of the manipulator with separate neural network compensators along the force and motion controlled directions in the task co-ordinate system. With the learning mechanism in the task space, modelling errors, dynamic friction and changes in environment stiffness are automatically compensated for which result in highly desirable task oriented performance characteristics. Simulation results are provided using the PUMA 560 arm which demonstrates the applicability of the proposed method to the position / force hybrid control of manipulators.

1. Introduction

Most present day industrial robots are used for welding, spray painting and material handling tasks which only require the control of manipulator end effector position and velocity. In contrast to these several applications such as assembly and machining of parts involve continuous contact between the end effector and the environment. In these applications the robot end effector applies forces on various objects in it's environment and moves along their surfaces. If all the parameters of the robot and it's environment are known and robot positioning is precise it might be possible to accomplish these tasks using motion control strategies only. However, in the presence of inevitable discrepancies in modelling the manipulator and the environment large reaction forces at the point of contact between end effector and the environment may be developed that might damage the manipulator and it's environment. In such situations controlling the interacting forces is a better option.

Conceptually, when a robot makes contact with a smooth and stiff surface, it is impossible to move the end effector into the surface or to exert a force tangent to the surface. In a properly defined task co-ordinate system the position controlled axes and the force controlled axes are orthogonal [9] thereby providing for independent control of position and force. Numerous position / force control schemes have been devised based on Cartesian co-ordinates of the end point or of some external reference frame. Overview of these schemes may be found in [17] [18]. These force / position control

schemes may be categorised in terms of whether a dynamic model of the arm is included in the control structure [1] [7] or not [11] [12].

The improvement in the accuracy and stability of the algorithms by using dynamic models of the arm in Cartesian force control has been demonstrated in [1]. Fixed gain position / force controllers not incorporating the system dynamics may provide reduced performance and have limited applicability. In the presence of a well developed dynamic model of the arm it is feasible to use it as a non-linear control input to cancel the coupling effects among joints and compensate for the inertial and gravitational forces. However, uncertainties can always exist in robot, sensor or environment models which if not neutralised may still degrade system performance.

Adaptive techniques have been proposed to improve the performance under parameter uncertainties [2] [3]. These approaches have been designed to maintain accurate motion and force control in the presence of structured uncertainties in system model, however, their performance cannot be guaranteed in the presence of unstructured uncertainties. If differences occur in the actual system and model structure robustness of the adaptive control schemes decreases [10]. Consequently, it may be desirable to utilise control techniques that learn from experience. Neural Networks offer one such alternative. Advances in artificial neural networks have provided the potential for new approaches to the control of systems with complex, unknown and non-linear dynamics. The advantages of using neural networks for control applications can be summarised as three-fold. First, they have a flexible structure to express non-linear systems which may provide for a robust controller. Second, because of their structure they have a flexible learning capability as against adaptive control method which stays within unknown parameter identification of a pre-defined model. In addition parallel processing and fault tolerance are easily achieved.

Recently there has been considerable interest in the application of neural networks to hybrid force position control of manipulators [5], [15], [16]. These efforts aim at compensating the dynamics of the arm and the environment within a central neural controller. Single link or two link manipulators have been considered in the above approaches. As the number of links of the arm increases so does the complexity and non-linearity of the underlying system which has to be controlled by the central neural controller. The neural network controller may therefore require excessive amount of training data in order to yield a reasonable generalisation and thereby increasing the

training period. As the inverse dynamics of manipulators can be computed efficiently to a fair degree of accuracy using Newton Euler equations, it is reasonable to utilise the inverse dynamic model and assist it through neural networks to compensate for the uncertainties of manipulators. Use of a model also decouples the position and force control directions so that independent neural networks of relatively small sizes can be used, each of which has to learn lesser amount of data as compared to a central neural controller.

This paper presents a de-centralised approach to adaptive hybrid control of robots using neural network compensators. The motion and force degrees of freedom are effectively decoupled through an inverse model of the robot. Each motion and force degree of freedom is then controlled separately by a neural network controller. The effectiveness of the scheme is demonstrated by simulation studies on PUMA 560 interacting with a rough surface in the presence of uncertainties in the robot model and that of the environment.

2. Dynamic model formulation

The dynamic equation of a general n-link manipulator can be represented by

$$D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + g(\theta) = u(t) \qquad (1)$$

where $u(t)$ is an $n\times1$ applied torque vector for junction actuators, $\theta(t)$ is the $n\times1$ joint angular position vector, $\dot{\theta}(t)$ is the $n\times1$ joint angular velocity vector, $\ddot{\theta}(t)$ is the $n\times1$ joint angular acceleration vector, $g(\theta)$ is an $n\times1$ gravitational force vector, $h(\theta,\dot{\theta})$ is an $n\times1$ coriolis and centrifugal force vector and $D(\theta)$ is an $n\times n$ acceleration-related inertia matrix. When the robot makes contact with the environment a reactive force at the end effector will be felt at each joint. The dynamic equation then becomes

$$D(\theta)\ddot{\theta} + h(\theta,\dot{\theta}) + g(\theta) = u(t) - J^{T}(\theta)\times R \qquad (2)$$

where $J(\theta)$ is the $n\times n$ Jacobian matrix, T is the transpose operator and R the reaction force / moment vector at the end effector. In this paper it is assumed that $J(\theta)$ is bounded away from singularities or equivalently that the arm is bounded away from the corresponding joint positions producing $J(\theta)$ to be singular. In order to perform hybrid control in Cartesian co-ordinates the dynamics can be derived in Cartesian co-ordinates as shown below.

Let x be an $n \times 1$ vector representing a set of Cartesian states of the end effector in a reference frame R_0. The forward kinematic relationship between joint positions and end effector configuration can be written as

$$x = f(\theta) \tag{3}$$

and the corresponding velocity relation

$$\dot{x} = J(\theta)\dot{\theta} \tag{4}$$

Differentiating equation (4) with respect to time the joint acceleration can be obtained

$$\ddot{\theta} = J^{-1}(\theta)(\ddot{x} - \dot{J}(\theta,\dot{\theta}) \times \dot{\theta}) \tag{5}$$

The applied torque vector u(t) in (2) can be replaced by a $n \times 1$ fictitious force / torque vector F using the relationship

$$u(t) = J^{T}(\theta) \times F \tag{6}$$

putting (5), (6) in (2) we get

$$D_x(\theta)\ddot{x} + H_x(\theta,\dot{\theta}) + G_x(\theta) + R = F \tag{7}$$

where $D_x(\theta)$ the Cartesian mass matrix, $H_x(\theta,\dot{\theta})$ the vector of velocity terms in Cartesian space and $G_x(\theta)$ the vector of gravity terms in Cartesian space are given by

$$D_x(\theta) = J^{-T}(\theta)D(\theta)J^{-1}(\theta)$$

$$H_x(\theta,\dot{\theta}) = J^{-T}(\theta)(h(\theta,\dot{\theta}) - D(\theta)J^{-1}(\theta)\dot{J}(\theta,\dot{\theta})\dot{\theta}) \tag{8}$$

$$G_x(\theta) = J^{-T}(\theta)g(\theta)$$

If the manipulator is in contact with a surface at a point whose position is x_e in R_0 frame and the effector position is x(t) then interaction can be modelled as a pure elastic restoring force / torque given by

$$R = K_e(x(t) - x_e) \tag{9}$$

where K_e is the environment stiffness matrix. If the manipulator is moving along a rough surface in a direction perpendicular to the force controlled direction, frictional forces are developed tangent to the surface which have not been considered in (7). The magnitude of these forces depends on the magnitude of reaction force R. In this paper the friction model used is

$$f_r = -(\mu R_i v + \zeta_{ran}) \tag{10}$$

where μ is the coefficient of friction, v is the unit vector in the direction of constrained velocity, R_i is the force applied on the surface and ζ_{ran} is a random signal due to surface irregularities. This friction force may be considered as an unknown disturbance which the position controller has to neutralise.

3. Hybrid Control

Using [4] and knowing the estimates of $D_x(\theta)$, $H_x(\theta,\dot{\theta})$ and $G_x(\theta)$, equation (7) can be used to decouple the constrained system into n sub-systems along n degrees of freedom in Cartesian co-ordinates and modelling errors can then be compensated by independent controllers with one controller along each degree of freedom. However, Cartesian space dynamic equation (7) is computationally very expensive which presents a major obstacle in its real time implementation. As an alternative to (7), it can be readily seen that the joint torque u(t) needed for the constrained motion and force control can be computed efficiently through Newton Euler equations using the formula

$$u(t) = D(\theta)\ddot{\theta}^* + h(\theta,\dot{\theta}) + g(\theta) + R^* \tag{11}$$

where $\ddot{\theta}^*$ is the $n \times 1$ acceleration command vector for motion control and R^* is the $n \times 1$ command vector for force control, both in joint co-ordinates. In this paper equation (11) is used for hybrid position / force control and $\ddot{\theta}^*$ and R^* are generated utilising neural networks to maintain position and force tracking accuracy in the presence of modelling errors.

3.1 Motion Control

For generating $\ddot{\theta}^*$, the motion command vector in joint space, first a pseudo-control vector is formed in Cartesian space utilising neural networks, which is then transformed into joint space using Jacobian relation. For p motion degrees of freedom $\ddot{\theta}^*$ is given below

$$\ddot{\theta}^* = J^{-1}\begin{bmatrix} NN^{(1)} + PD^{(1)} \\ \vdots \\ NN^{(p)} + PD^{(p)} \end{bmatrix}$$

$$= J^{-1}\begin{bmatrix} f_1(\theta,\dot{\theta},\ddot{x}_d^{(1)}) + \{K_p(x_d^{(1)} - x^{(1)}) + K_v(\dot{x}_d^{(1)} - \dot{x}^{(1)})\} \\ \vdots \\ f_p(\theta,\dot{\theta},\ddot{x}_d^{(p)}) + \{K_p(x_d^{(p)} - x^{(p)}) + K_v(\dot{x}_d^{(p)} - \dot{x}^{(p)})\} \end{bmatrix} \tag{12}$$

where $NN^{(i)} = f_i(\theta, \dot{\theta}, \ddot{x}_d^{(i)})$ is the non-linear compensating function of joint positions, joint velocities and desired Cartesian acceleration along the ith degree of motion freedom, implemented through neural networks and $PD^{(i)} = K_p(\cdot) + K_v(\cdot)$ is the output of the PD controller in Cartesian co-ordinates along the ith degree of motion freedom with K_p and K_v being positive gains. The PD controller output is also an error signal for updating the weights of the corresponding neural network. Each of the p neural networks has 2n+1 inputs and one output. Equation (12) may be compared with (5) which may be written as

$$\ddot{\theta} = J^{-1}\begin{bmatrix} g_1(\theta, \dot{\theta}, \ddot{x}^{(1)}) \\ \vdots \\ g_p(\theta, \dot{\theta}, \ddot{x}^{(p)}) \end{bmatrix} \tag{13}$$

The job of neural network implementing $f_i(\cdot)$, apart from compensating for $g_i(\cdot)$, is to reduce motion tracking errors along the ith degree of motion freedom, while motion errors along the remaining n-p force control directions are ignored.

3.2 Force control

Similar to the motion control strategy pseudo-control vectors are generated in Cartesian co-ordinates in the n-p force degrees of freedom which are then transformed into joint co-ordinates using Jacobian relations to form R^*. When contact is made with a hard surface having a large stiffness value, due to very high loop gains, the force error signal $f_e' = f_d - f$ has very high frequency components. Use of this signal as the error measure for updating the weights of neural networks in the force loop resulted in poor convergence. This force error signal was smoothed out to generate f_e, a suitable error measure for the neural networks in force loops. Smoothing was performed using an exponentially weighted recursive filter given by the following equation

$$f_e^{(k+1)} = \alpha \times f_e^{(k)} + (1-\alpha) \times (f_d^{(k+1)} - f^{(k+1)}) \qquad 0 \le \alpha \le 1 \tag{14}$$

The force command vector R^* is then given by

$$R^* = J^T\begin{bmatrix} NN^{(1)} + P^{(1)} \\ \vdots \\ NN^{(n-p)} + P^{(n-p)} \end{bmatrix}$$

$$R^* = J^T \begin{bmatrix} F_1(f_d^{(1)}, f^{(1)}) + K_f f_e^{(1)} \\ \cdot \\ \cdot \\ F_{(n-p)}(f_d^{(n-p)}, f^{(n-p)}) + K_f f_e^{(n-p)} \end{bmatrix} \tag{15}$$

which can be compared to (6). $NN^{(i)} = F_i(f_d^{(i)}, f^{(i)})$ is the non-linear function compensator implemented by the neural network along the ith degree of force freedom and $P^{(i)} = K_f(\cdot)$ is the proportional gain also used as an error signal for neural network weight update. Each neural network in the force loop has two inputs and one output. Force errors along the p motion controlled directions are ignored.

The control structure for an n link robot is shown in Figure 1 where EWRF is the exponentially weighted recursive filter and Γ represents the transformations from joint space to Cartesian space. The scheme is computationally efficient and is amenable to parallel processing implementation within a computing architecture. The inverse dynamics part, being computationally the most involved part in the algorithm can be computed using Newton Euler equations in time linear in the number of links on a single processor. Moreover very efficient implementations on Newton Euler formulas have been achieved on multiple general purpose processors, also, pipelined dedicated hardwares have been proposed to further cut down the computation time [6] [8].

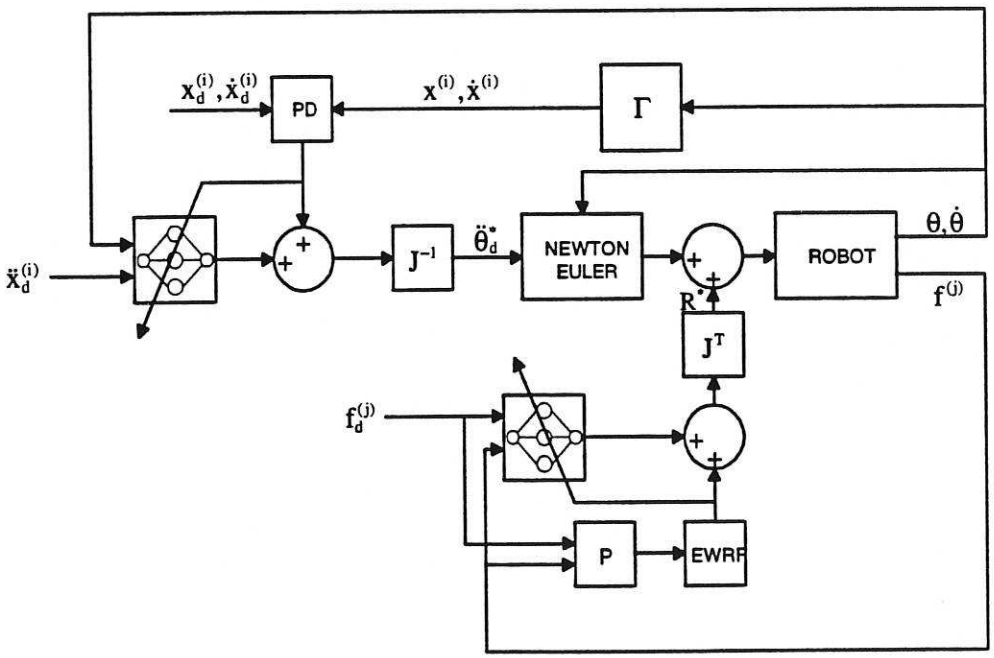

Figure 1. Hybrid control structure

4. Simulation application

To demonstrate the effectiveness of the control scheme , first 3 links of PUMA 560 arm are used in dynamic simulations [13] [14]. The schematic diagram is shown in Figure 2. A global reference frame R_0 is fixed to the base frame. The manipulator end point is moved in a circular trajectory along a surface parallel to y-z plane of R_0 while applying a force in the direction of x axis. A rigid robot is assumed. Modelling errors are introduced by changing the values of robot inertia parameters by 25% from those used in the controller design, also, using equation (10), frictional forces acting as disturbances for the motion control part were introduced tangent to the surface. There are two neural network compensators for the two motion degrees of freedom and one neural network in the single force loop. Every neural network has two hidden layers with sigmoidal activation function and a linear output neuron. The number of neurons in the first and second hidden layer being 20 and 10 respectively. Standard back-propagation is employed. The structure of the neural networks is given in Figure 3 and simulation parameters are given in Table 1. Two values for the environment stiffness were used, 10^7 N/m for hard surfaces and 10^4 N/m for relatively soft surfaces. For the sake of comparison force control without neural network control is also shown.

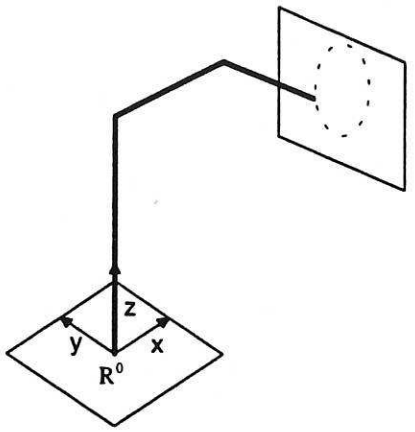

Figure 2. schematic diagram of 3 degree of freedom constrained system.

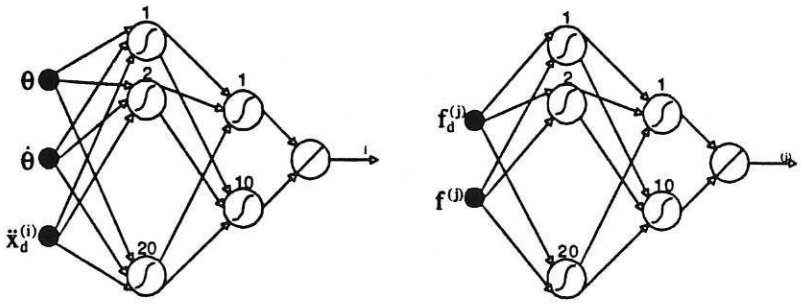

Figure 3. Neural network structure used in the position and force loops.

Table 1. Simulation parameters.

K_p	500
K_v	50
K_f	0.1
μ	0.2
α	0.98
Inertia parameter error	25 %
Control cycle	200 Hz
Weight update rate	200 Hz

Neural network weights were initialised to small random values. Figure 4 shows the desired and actual motion trajectories along the y-z plane. Very low tracking errors were observed through out the course of the circular trajectory. Figure 5 shows the desired and actual force trajectories applied in the direction of x axis while moving along the y-z plane. Typical force trajectories are constant values, however, to evaluate the control scheme a trajectory containing ramp and step elements is shown. Figure 5(a) represents the force trajectory with the value of environment stiffness equal to 10^4 N/m, while Figure 5(b) represents the same with stiffness equal to 10^7 N/m. Smooth contact may be observed in Figure 5 (a), while in Figure 5(b), because of very high force loop gain, some oscillations are observed at the time of making of contact, with a maximum value close to twice the desired value. These oscillations, however, die down rapidly and once the initial oscillations are over, accurate force tracking for both the stiffness values may be observed. Figure 6(a) and 6(b) depict the components of forces of friction along the y and z axes respectively, that act as disturbances for the motion controllers. To have a comparison, force control without neural networks is plotted in Figure 7. In Figure 7(a) the value of stiffness is 10^4 N/m, while in Figure 7(b) it's value is 10^7 N/m. Apart from increased oscillations at the time of making contact, it may be observed that tracking is particularly poor for the higher value of stiffness and is accompanied with large oscillations. Figure 8 displays the processed force error, which is the output of the recursive filter that was used to update the weights of the neural network in the force loop.

In the next stage, robustness of the controller against constraining surface movement along the force direction is being demonstrated. The constraining surface is given a velocity in the positive x direction. It should be noted that position is not being controlled in the x direction. The robot end point was made to follow a circular trajectory similar to that of Figure 4 along the y-z plane and was required to keep the force in the x direction constant at 20 N, despite surface movement in that direction. Simulation results for surface movement in the force controlled direction are given in Figures 9, 10 and 11. Figure 9 (a) shows the force response for the moving surface having a stiffness of 10^4 N/m. and a constant velocity of 2 cm/sec. in the positive x direction. In this case, larger oscillations that persist for a longer duration of time, are observed at the start, when compared with the fixed surface case of Figure 5. However, once the neural network gets trained it is then able to maintain the force near the desired value, despite surface movement. Figure 9 (b) shows a similar force response but with surface stiffness equal to 10^7 N/m. and surface velocity reduced to 2 mm./sec. It may be noted that for lower surface stiffness the manipulator end point was able to track and maintain the desired force against a much faster moving surface. Movement of the surface is shown in Figures 10 and 11, for stiffness values of 10^4 N/m. and 10^7 N/m. respectively. Figure 10 (a) and 11 (a) show the position of the surface along the x axis, while Figure 10 (b) and 11 (b) show the x co-ordinate of end point of the manipulator as it tracks the surface while applying the desired force. Apart from small errors at the start the end point follows the surface closely.

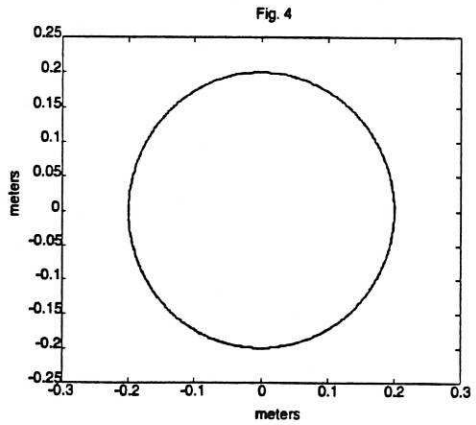

Figure 4: desired and actual position trajectory along the y-z plane.

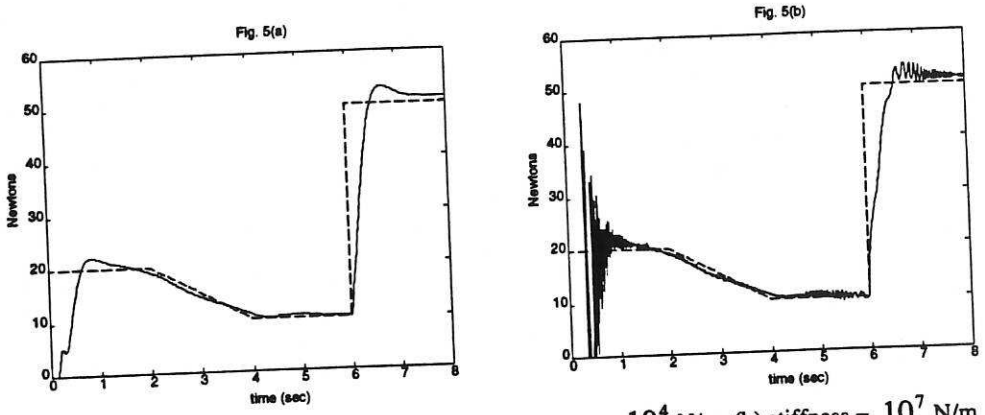

Figure 5: desired and actual force trajectories. (a) stiffness = 10^4 N/m. (b) stiffness = 10^7 N/m.

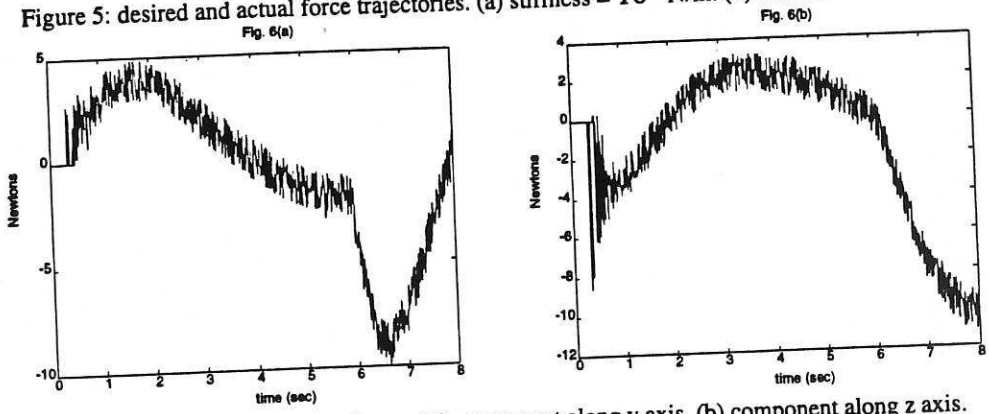

Figure 6. disturbance friction forces. (a) component along y axis. (b) component along z axis.

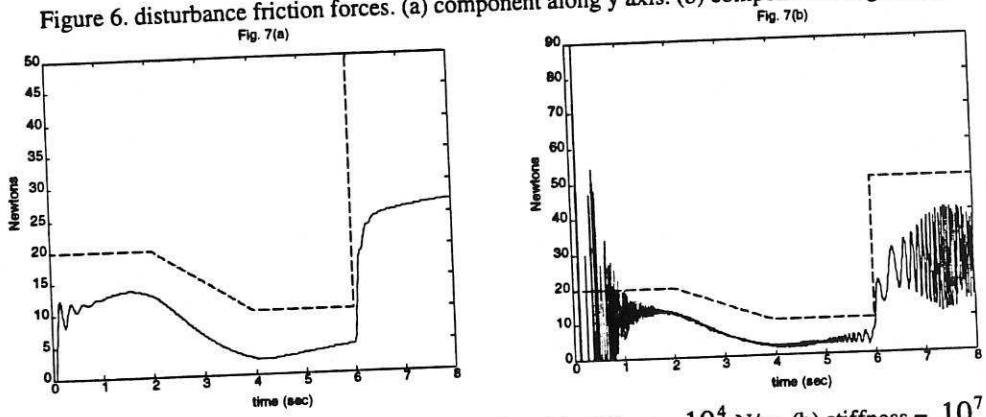

Figure 7: Force trajectories without neural compensation. (a) stiffness = 10^4 N/m. (b) stiffness = 10^7 N/m.

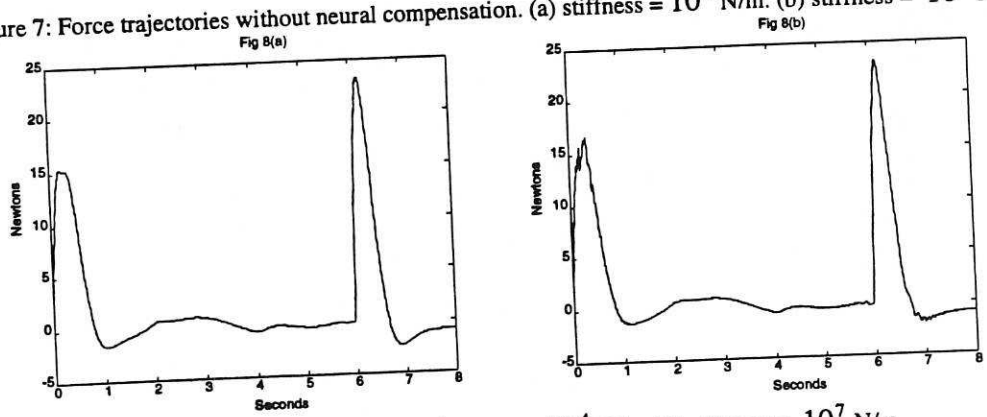

Figure 8: Filtered force error (a) stiffness = 10^4 N/m. (b) stiffness = 10^7 N/m.

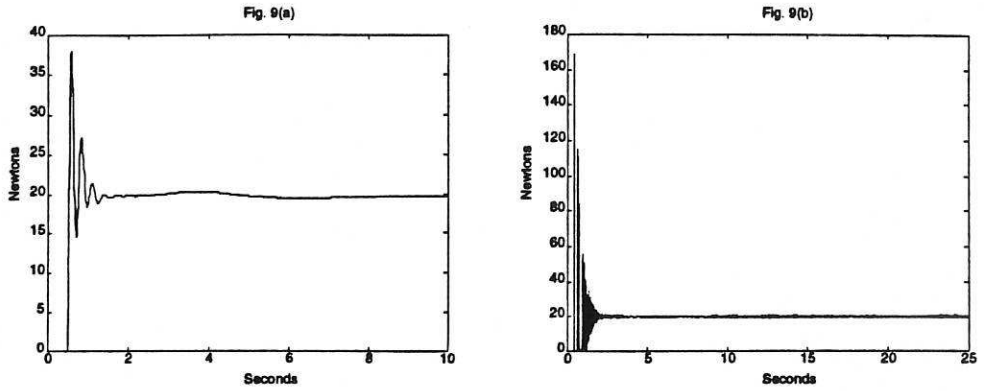

Figure 9: Force trajectories against moving surface. (a) stiffness = 10^4 N/m. (b) stiffness = 10^7 N/m.

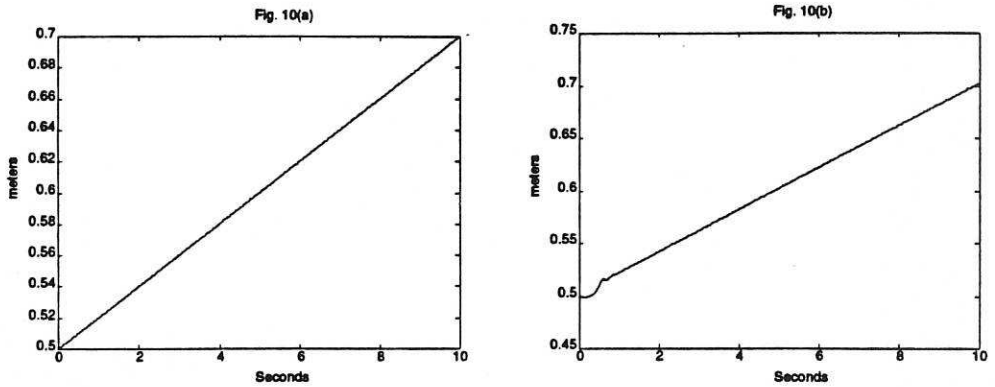

Figure 10 (a): Constraint surface movement, (b): endpoint tracking movement. (stiffness = 10^4 N/m.)

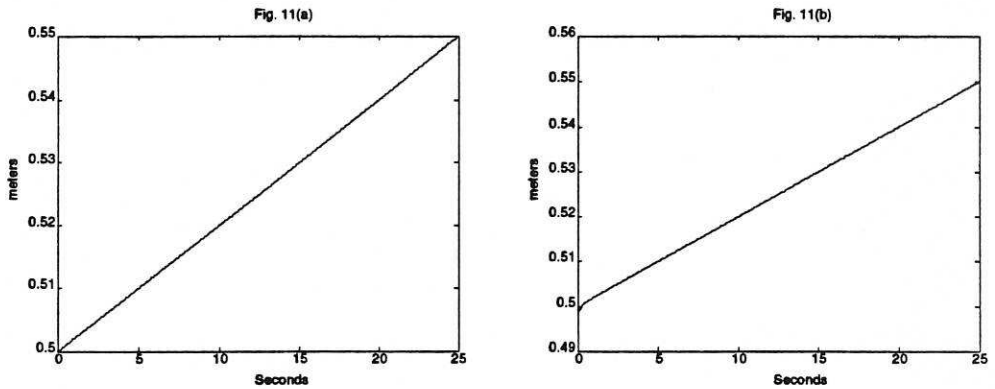

Figure 10 (a): Constraint surface movement, (b): endpoint tracking movement. (stiffness = 10^7 N/m.)

The above figures show that despite surface movement the hybrid controller was able to maintain the desired force.

5. Conclusions

This paper presents a neural network based scheme for hybrid position / force control of manipulators. The scheme is based on the belief that neural networks perform best when they are not required to learn too much data, too fast. Use of nominal dynamic model of manipulators in conjunction with neural networks reduces the job of neural networks to compensation of modelling uncertainties rather than generating the model itself. Simulation studies conducted on a realistic system, demonstrate accurate

position and force tracking performance in the presence of uncertainties in the manipulator model and the environment. The scheme is computationally efficient and is well suited for parallel processing implementation within a computing architecture.

References

1] An C. H. and Hollerbach J. M., "The role of dynamic models in Cartesian force control of manipulators", *Int. J. of Robotics Research*, Vol. 8 No. 4, (1989).

2] Carelli R., Kelly R. and Ortega R., "Adaptive force control of robot manipulators", *Int. J. Control*, Vol. 52, No. 1, (1990).

3] Chung J. C. H. and Leininger G. G., "Task-level adaptive hybrid manipulator control", *Int. J. of Robotics Research*, Vol. 9, No. 3, (1990).

4] Freund E., Fast nonlinear control with arbitrary pole-placement for industrial robots and manipulators. *Int. J. Robotics Research*, Vol. 1, No. 1, (1982).

5] Fukuda T., Kurihara T., Shibata T., Tokita M. and Mitsuoka T., "Applications of neural network-based servo controller to position, force and stabbing control by robotic manipulator", *JSME Int. J. Series 3*, Vol. 34, No. 2, (1991).

6] Kasahara H., "Parallel processing of robot arm dynamic control computation on multiprocessors," *Microprocessors and Microsystems*, Vol. 14 No. 1, (1990).

7] Khatib O., "A unified approach for motion and force control of robot manipulators: The operational space formulation", *IEEE J. Robot. Automat.*, Vol. RA-3, No. 1, (1987).

8] Lathrop R. H., "Parallism in arms and legs," MIT, M. Sc. thesis, (1982).

9] Mason M. T., "Compliance and force control for computer controlled manipulators", *IEEE Trans. on System Man and Cybernatics*, Vol. SMC-11, No. 6, (1981).

10] Pao Y. H., Phillips S. M. and Sobajic D. J., "Neural-net computing and the intelligent control of systems", *Int. J. Control*, Vol. 56, No. 2, (1992).

11] Railbert M. H. and Craig J. J., "Hybrid position / force control of manipulators", *Trans. of the ASME J. of Dynamic Systems, Measurement, and Control,* Vol. 102 June (1990).

12] Salisbury J. K., "Active stiffness control of a manipulator in Cartesian coordinates", Proc. 19th IEEE Conf. Decision and Control, pp 95-100, (1980).

13] Tarn T. J., Bejczy A. K., Han S. and Yun X., Dynamic equations for puma 560 robot arm. Robotics Lab. Report SSM-RL-85-02, Washington University, St. Louis, Missouri, (1985).

14] Tarn T. J., Bejczy A. K., Han S. and Yun X., Inertia parameters of puma 560 robot arm. Robotics Lab. Report SSM-RL-85-02, Washington University, St. Louis, Missouri, (1985).

15] Tokita M., Mituoka T., Fukuda T., Shibata T. and Arai F., "Position and force hybrid control of robotic manipulator by neural network", *IEEE International Joint Conference on Neural Networks*, Vol. 1, pp 113-21, (1991).

16] Tokita M., Mituoka T., Fukuda T. and Kurihara T., "Force control of robotic manipulator by application of a neural network", *Advanced Robotics*, Vol. 5, No. 1, (1991).

17] Vukobratovic′ M. and Tuneski A., "Contact control concepts in manipulation robotics-An overview", *IEEE Trans. on Industrial Electronics*, Vol. 41, No. 1, (1994).

18] Whitney D. E., "Historical perspective and state of the art in robot force control", *The Int. J. of Robotics Research*, Vol. 6, No. 1, (1987).

19] Widrow B. and Lehr M. A., 30 years of adaptive neural networks: perceptron, madaline, and backpropagation. *Proc. IEEE*, Vol. 78, No. 9, (1990).