A Neural Network Methodology For Engineering Applications: The Learning Control of Robotic Systems

Ali M. S. Zalzala

Robotics Research Group,
Department of Automatic Control and Systems Engineering,
University of Sheffield,
Mappin Street, P.O.Box 600, Sheffield S1 4DU, United Kingdom

Research Report #480
May 1993

Abstract

This paper presents a methodology for the application of artificial neural networks in engineering applications. In this approach, the conventional control model is used to provide knowledge to the designed network, hence giving a further insight into the solution of the specific task. The presented method is implemented for different modules of robot control with outstanding success.

1. Introduction

The application of the theory of artificial neural networks has emerged in recent years as a very attractive area of research. For the engineering community, the theory of cognition offers a possible solution for different inherent problems for which the massive parallelism and the learning abilities associated with neural networks are very much appreciated. Different approaches are reported in the literature for implementing neural-based control algorithms [1], including simple PID controllers [2] and the more sophisticated model-reference adaptive controller (MRAC) [3]. In addition, other applications in system identification [4] and nonlinear system theory [5] are also reported. Many successful solutions are delivered in the field of pattern recognition [6] where different network designs are available. However, once real-time dynamic systems are concerned, less success has been achieved [7].

The fast and accurate control of articulated robot manipulators forms a difficult challenge to engineering researchers due to the very complicated nature of the overall dynamic system [8]. Problems in the control of robots arise from the vast computational complexities associated in its mathematical formulation, in addition to the need for appropriate adaptive control methods to achieve the required precision and speed [9]. Therefore, the robotics community has been interested in connectionism as one possible solution, where the massive parallelism within the network can reduce the computational burden. In addition the learning abilities make such approach attractive in the sense of providing a compensation for any errors in executing the robot motion. Thus, by producing a robot control model based on the structure of the human brain, it is hoped to inherit the latter abilities and achieve a human-like behaviour [10].

2. Robot Control Modules

A typical robot manipulator consists of a series of rigid links connected to each other by either revolute or translational joints, and the position of the robot hand can be expressed in the cartesian (3D) space or the joint space. The general problem of robot motion control requires moving the robot hand from one point in space to another. All joints on the arm are related by assigning a coordinate frame to each, in addition to certain link parameters. Consequently, general robot control considerations are defined as follows [11]

(1) Robot Kinematics: deals with the geometry and motion of the robot without considering the forces causing the motion. The *direct* kinematic equations transform

the joint values to their cartesian equivalent, while the *inverse* kinematics relates in the opposite direction.

(2) Robot Jacobian: considers beyond the static positioning mentioned in (1) above to include linear and angular velocities. The notations for inverse and direct jacobian is the same as above.

(3) Robot Dynamics: takes into account the forces required to cause the motion and their effect on the system behaviour. The *inverse dynamics* formulation computes the required torque (force) values given the desired parameters of motion (i.e. position, velocity and acceleration), while the direct dynamics computes the reverse.

The following control modules are formulated for the Unimation PUMA 560 manipulator with six revolute joints.

2.1. *Inverse Jacobian Formulation*

The most computationally efficient formulation for the inverse jacobian for a manipulator with a spherical wrist is given as [12]

$$\dot{\theta} = J^{-1}\dot{\chi} = \begin{bmatrix} J_{11} & 0 \\ J_{21} & J_{22} \end{bmatrix}^{-1} \dot{\chi}, \quad J^{-1} = \begin{bmatrix} J_{11}^{-1} & 0 \\ -J_{22}^{-1}J_{21}J_{11}^{-1} & J_{22}^{-1} \end{bmatrix}$$

where $\theta = [\theta_1,\theta_2,\theta_3,\theta_4,\theta_5,\theta_6]^T$ and $\chi = [P_x,P_y,P_z,D_x,D_y,D_z]^T$ are the joint rates and the cartesian rates, respectively and $\mathbf{J}^{-1}$ is the inverse jacobian matrix, and various elements of **J** are defined in a semi-symbolic form [17].

2.2. *Inverse Dynamics Formulation*

The Lagrange-Euler dynamic model of the robot computes the torque values τ as

$$\tau_i = \sum_{j=1}^{N} D_{ij}\ddot{\theta}_j + \sum_{j=1}^{N}\sum_{k=1}^{N} H_{ijk}\dot{\theta}_j\dot{\theta}_k + C_i\,, \quad i=1,..,no.\ of\ joints(N)$$

where θ,θ,θ are the motion parameters, and matrices $\mathbf{D}_{ij},\mathbf{H}_{ijk},\mathbf{C}_i$ are the effective and coupling inertias, centripetal and coriolis forces and gravity loading, respectively. Expanding individual terms of the above equation yields a computationally efficient symbolic representation [18].

3. Neural Application Drawbacks

The general form of the multi-layered neural network uses the neuron as its basic block [13]. The fact that the biological origin on which such a neural model is based is still being investigated by neurobiologists [14] is of no real concern to engineering researchers. As far as control systems are concerned, certain concepts of neural networks are readily adapted by engineers as presented to them, and attempts are made to use the rather ambiguous model to solve a required problem. However, such approach introduces enormous application difficulties as discussed below.

3.1. *Constructing the Network*

A typical neural network model consists of an input layer, and output layer and a number of hidden layers of neurons. As illustrated in figure (1), the hidden part of the network can be considered as a black box for which a suitable design must be introduced, preferably to imitate the brain capabilities. Therefore, to apply the required engineering theory, certain assumptions are made, including setting up some number of hidden layers each containing a number of computational neurons. In addition, learning parameters must be given which depends on the learning algorithm used by the network. Once the network design is finished, execution will commence by applying the desired inputs and further checking the actual output against the desired values, where the network is expected to learn from the errors. Hence, this execution procedure is repeated iteratively for hundreds or even thousands of times during which it is hoped that convergence may occur.

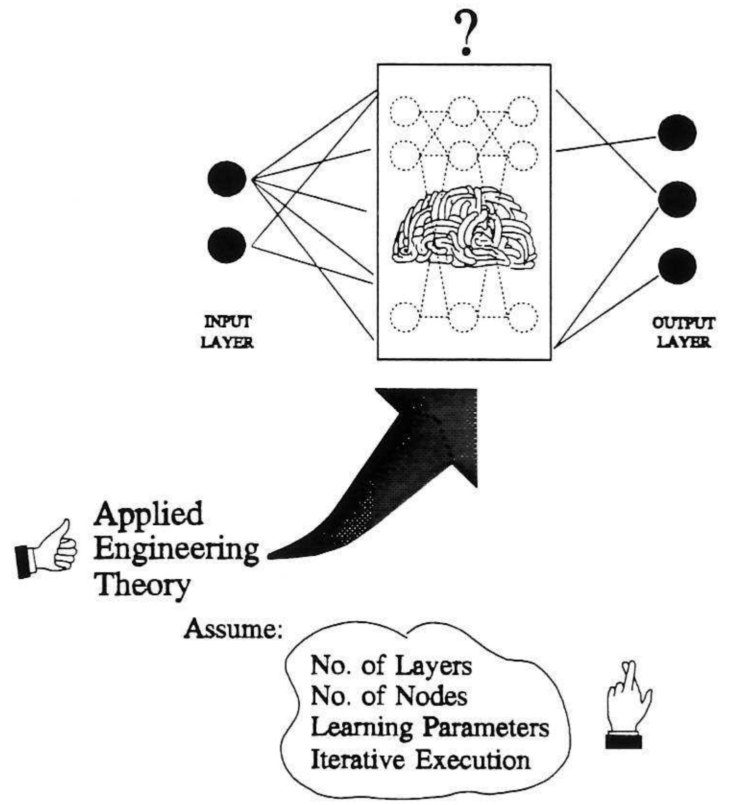

Figure (1): Engineering Approach to Neural Applications

(1) Since the combining functions are different for each neuron, and each requiring a different set of inputs, the network will have sparse connections. In addition, some neurons in a hidden layer may have connections not only from the immediate previous layer but from other previous ones as well.

(2) Due to having sparse connections, the network does not suffer from the so-called problem of symmetry [13], and initial setting of weights can have equal values (e.g. unity).

(3) The computational burden associated with each neuron will depend on the complexity of the combining function derived from the system model.

Considering the above procedure, a parallel machine can be organised, where all neurons and every connection is accounted for according to the original system model. In this approach, no ambiguities are tolerated when designing the network.

5. Implementation

The general structure of the neural controller is shown in figure (2), where **I** and **O** are the inputs and outputs, respectively, and subscripts *a* and *d* denote the actual and desired parameters, respectively. Thus, using the desired input parameters, the neural controller produces a set of control commands moving the robot arm to a certain actual position. A feedback of this actual position is used by the computational module (which is similar to the controller but with no learning abilities) producing a set of actual control commands. Finally, the difference between the actual and desired control commands is used to adjust the neural network by propagating the errors throughout its layers. Using the above procedure, two robot modules are implemented, namely the inverse jacobian and the inverse dynamics.

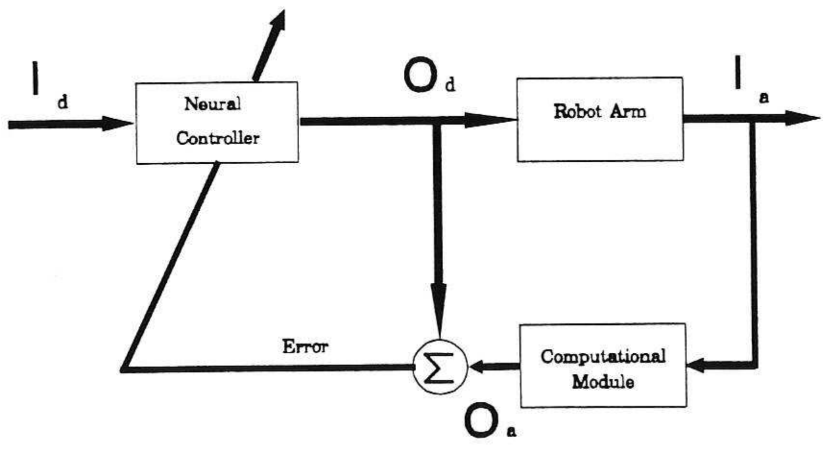

Figure (2): Neural Controller Structure

5.1 *Neural Inverse Jacobian (NIJ)*

Considering the formulation of section 2.1, the input **I** is set to χ and the output **O** are set to θ. The network has six hidden layers as shown in figure (3), where the number of neurons in each is shown totalling 66 nodes. Applying the NIJ using a practical motion trajectory while introducing some imposed deviations [17,18] lead the controller to reduce the tracking errors for all six joints of the arm, as shown in table (1). According to table (1), all joint errors were reduced significantly except for joints 4 and 6, which is attributed to being caught in a local minima.

Tracking Errors (degrees)	PUMA Joint					
	1	2	3	4	5	6
No Learning	0.511	0.159	0.164	0.264	0.457	0.251
With Learning	0.079	0.052	0.052	0.251	0.065	0.247

Table (1): Reducing the errors by the NIJ

5.2 *Neural Inverse Dynamics (NID)*

The same approach is used to implement the inverse dynamics of section 2.2, which is far more computationally intense than the inverse jacobian [18]. A network with six hidden layers is considered using a number of 111 neurons for which $\mathbf{I}=\{\theta,\theta,\theta\}$ and $\mathbf{O}=\tau$, as shown in figure (4). Simulation results using the same motion trajectory of section 5.1 are shown in table (2).

Tracking Errors (degrees)	PUMA Joint					
	1	2	3	4	5	6
No Learning	4.823	2.566	3.626	55.93	64.38	45.16
With Learning	2.933	0.876	1.684	37.08	50.81	34.22

Table (2): Reducing the errors by the NID

6. Discussion and Conclusion

A new approach to neural robot control is presented, where conventional robot control theory is augmented with aspects of the theory of cognition. This approach gives the neural network some initial knowledge of the system model, where the network is expected to behave better once it knows something about its task. Such initial knowledge can be readily available in engineering applications where data of the controlled system is provided for, as it is the case in robotics.

The main aim of this approach is to make use of the parallelism for real-time implementation, in addition to the learning abilities to perform adaptive control. However, no model identification is attempted here since the robot model is already well defined, while the remaining task is efficiently controlling the arm. The implementations reported illustrates the practicality of the procedure for any robot structure with any number of joints. In addition,

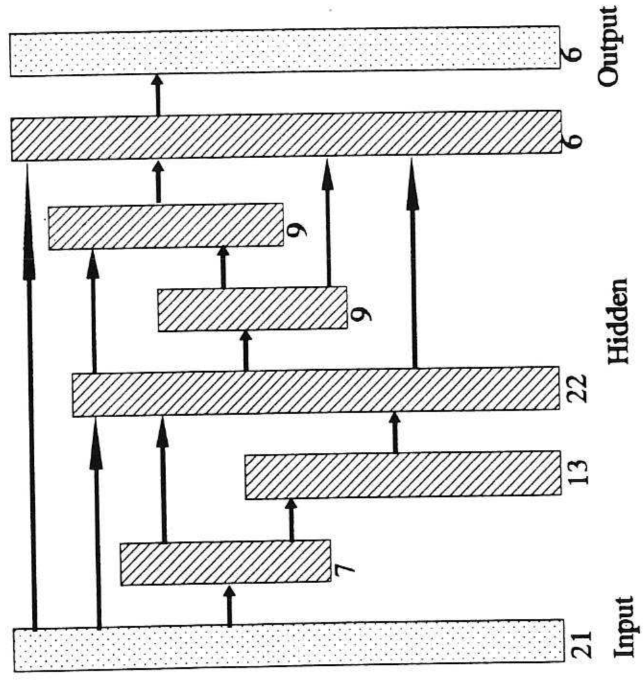

Figure (3): Neural Inverse Jacobian

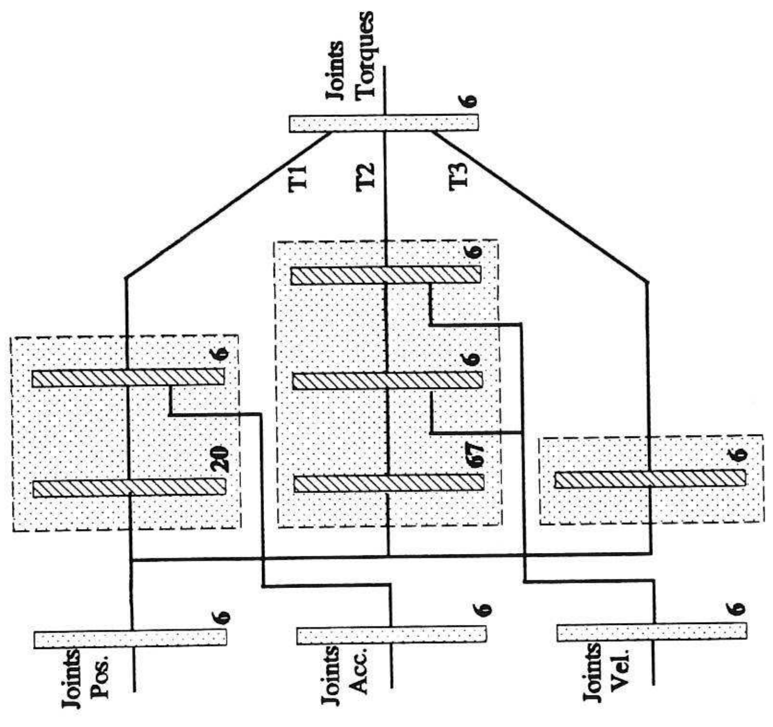

Figure (4): Neural Inverse Dynamics

no reservations are made for the type of the executed motion. Finally, although this implementation is reported for a robotic system, the same procedure can be applied to solve any problem where real-time dynamic systems are involved.

References

[1] Zbikowski, R. and Gawthrop, P.J., "A survey of neural networks for control", In Neural networks for control and systems, eds.: K. Warwick, G.W. Irwin, K.J. Hunt, IEE, London, 1992

[2] Ruano,A.E.B., Jones, D.I. and Fleming, P.J., "A neural network controller", In Proc. IFAC Workshop on Algorithms and Architectures for Real-Time Control, Bangor, UK, 1991

[3] Lightbody, G., Wu, Q.H. and Irwin, G.W., "Control applications for feed forward networks", In Neural networks for control and systems, eds.: K. Warwick, G.W. Irwin, K.J. Hunt, IEE, London, 1992

[4] Billings, S. and Chen, S., "Neural networks and system identification", In Neural networks for control and systems, eds.: K. Warwick, G.W. Irwin, K.J. Hunt, IEE, London, 1992

[5] Banks, S.P. and Harrison, R.F., "Can perceptrons find Lyapunov Function - An algorithmic approach to system stability", Report 365, Dept. of Control Engg., Sheffield Univ., UK, 1989

[6] Pao Y.-H., *Adaptive pattern recognition and neural networks*, Adison Wesley, 1989

[7] Berhen, J., Gulati, S. and Zak, M., "Neural learning of constrained nonlinear transformations", *IEEE Computer*, Vol. 22, No. 6, pp. 67-76, 1989

[8] Fu, K.S., Gonzalez, R.C. and Lee, C.S.G., *Robotics: Control, Sensing, Vision and Intelligence*, McGraw Hill, New York, 1987

[9] Roele, M. and Warwick, K., "Combining adaptive and neural control", In Proc. IFAC Workshop on Algorithms and Architectures for Real-Time Control, Bangor, UK, 1991

[10] Arbib, M.A., *The Metaphorical Brain 2: Neural Networks and Beyond*, Wiley, 1989

[11] Paul, R.P., *Robot Manipulators: Mathematics, Programming and Control*, MIT Press, 1981

[12] Paul, R.P. and Zhang, H., "Computationally efficient kinematics for manipulators with spherical wrists based on the homogeneous transformation representation", *Int. J. Robotics Research*, Vol. 5, No. 2, pp. 32-44, 1986

[13] Rumelhart, D.E. and McClelland, J.L., *Parallel Distributed Processing: Exploration in the Microstructure of Cognition*, Vol. 1,2, MIT Press, 1986

[14] Crick, F.H.C. and Asanuma, C., "Certain aspects of the anatomy and physiology of the cerebral cortex", In *ibid*, Vol. 2, pp. 333-40

[15] Blum, A.L. and Rivest, R.L., "Training a 3-node neural network is NP-complete", *Neural Networks*, Vol. 5, pp. 117-27, 1992

[16] Minsky, M.L. and Papert, S.A., "Epilog: the new connectionism", In *Perceptrons*, pp. 247-80, MIT Press, Third edition, 1988

[17] Zalzala, A.M.S., "Robot jacobian control: A new approach via artificial neural networks", In Proc. 1st. Int Conf. on Intelligent Systems Engineering, Edinburgh, 1992

[18] Zalzala, A.M.S. and Morris, A.S., "A neural network approach to adaptive robot control", *Int. J. Neural Networks*, Vol. 2, No. 1, pp. 17-35, 1991