Journal of Modern Technology & Engineering
Vol.3, No.2, 2018, pp.117-124

THE NEURAL NETWORK APPLICATIONS TO CONTROL OF ROBOT MANIPULATORS

Bekir Cirak[1*]

[1]Department of Mechanical Engineering, Engineering Faculty, Karamanoglu Mehmetbey University, Yunus Emre Campus, Karaman, Turkey

Abstract. In this study kinematics calculations have been mentioned so that a robot arm can reach the desired position. Dynamic calculations have been mentioned in the calculation of the required torque forces in the joints in order to reach the desired position. In addition, the ANN-based control method is used to estimate the parameters close to the desired parameters.

Keywords: *Neural networks, robots, motion control, kinematic, dynamic.*

Corresponding author: *Bekir Cirak, Department of Mechanical Engineering, Karamanoglu Mehmetbey University, Engineering Faculty, Yunus Emre Campus, Karaman , Turkey, tel: +9 0338 2262200, e-mail: bekircirak@mynet.com*

Received: *14 May 2018;* ***Accepted:*** *08 June 2018;* ***Published:*** *02 August 2018*

1. Introduction

Robotics is the science and engineering dealing with the design and application of reprogrammable multifunctional manipulators to move objects through variable programmed motions for the performance of a variety of tasks. Control of robots requires the solution of the complex inverse kinematic and dynamic equations, which is a computationally intensive process. Also the parameters of a robot such as moments of inertia and joint friction can not be determined precisely and obtaining meaningful model equations is difficult. Hence the idea of obtaining these relations based on measured inputoutput data is very appealing.Artificial neural networks can be thought of as a class of computational models for representing non-linear input-output mappings. Learning occurs by training with examples rather tha explicit programming. These properties make neural networks very attractive for robot control.

Neural networks consist of a very large number of simple processing elements called neurons, with each neuron connected to a large number of other neurons. The neurons are very simple by themselves, but the real power of neural networks lies in the interconnection between these simple neurons. The learning algorithms modify the strengths of these interconnections during a training phase. The desired mapping is thus encoded in these interconnections of a trained neural network to give immense parallel processing capabilities. Some of the important forms of neural network architectures used in robotics are the multilayer feedforward networks and the recurrent networks.The trajectory of a robot refers to the time history of position, velocity and acceleration for each degree of freedom. Trajectory generation involves the computation of motion functions such that each manipulator joint moves as a smooth function of time and the motion appears coordinated. A polynomial is generally fitted to a sequence of desired

points.The exact form of the required functions of actuator torque depend on the spatial and temporal attributes of the path taken by the end-effector as well as the mass properties of the links and payload, friction in the joints, etc. Study of robotics involves the study of following areas: trajectory generation, kinematics, dynamics and control(Abdelhameed, 2005)

2. Artificial neural networks

The study of human brain functions triggered the development of the science of neural networks. It was Neuroscience that first attempted to explain the way that the human brain works on the basis of simple mathematical models. A neural network is composed of large numbers of highly interconnected processing elements known as neurons. The basic elements of an artificial neuron are shown in Fig 1.

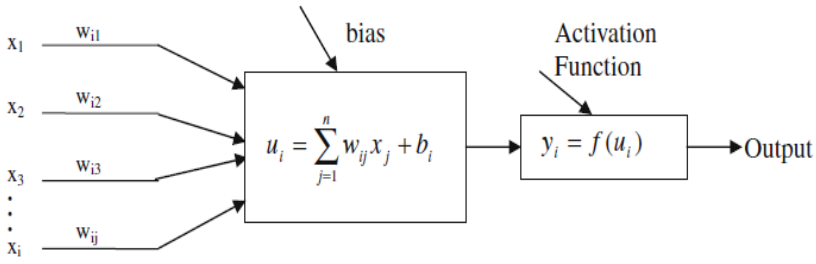

Figure 1. The style of neural computation

An ANN consists of artificial neuronsinspired from biological neurons. Artificial neurons are typically organized in layers, so that each ANN includes:

An input layer: for input data. This layer has as many neurons as the ANN's input variants. Input layer neurons are connected to the neurons of the hidden layers or to neurons in the next layer.

Hidden layers: Each hidden layer can have n neurons linked in different ways to the other hidden layers or tothe output layer. Hidden layer neurons can get their input through the input layer or some other hidden layer or in some cases, even the output layer.

An output layer: through which the output vector passes. This layer has as many neurons as the ANN's output variants. Output layer neurons can get their input through the input layer or the hidden layers.

The main feature of ANNs is their inherent capacity to learn, a key component of their intelligence. Learning is achieved through *training,* a repetitive process of gradual adaptation of the network's parameters to the valuesrequired to solve a problem. There are three training methods: supervised training, unsupervised training and reinforced training. The most common training method is supervised ining. An ANN is an information processing system consisting of simple processing elements, called artificial neurons. Therefore, it is essential to understand the function of an artificialneuron, which involves the following components:

The *input signals* X_j or *input information.*

The *synapses*, which are accompanied by a *synaptic weight*. Each input signal X_j at the synapsis entry, which is connected to the neuron *k*, is multiplied with its respective

weight Wk_j. The *summing junction* or *adder* Σ of the input signals, after they have been adapted with the use of their synaptic weights.

The activation or transfer function φ(.) To limit the neuron's output range in a closed unit interval [0,1] or [-1,1]. Any function can be used as activation function, and each ANN can have neurons with different function. There are various types of activation functions, including linear function, piecewise-linear function, step function, stochastic function, sigmoidal function etc.

The output signal *yk*, which is essentially the result produced by the artificial neuron k. It is also referred to as the artificial neuron's actual response (Cirak & Demirtas, 2014).

Finally, the artificial neuron includes a term that is applied externally. This external term helps prevent errors in cases of zero input data. The above simplified structure of an artificial neuron can bemathematically expressed by equations (1) and (2).

$$U_k = \sum_{j=1}^{n} W_k . X_j + b_k \quad (1)$$

$$Y_k = \varphi(U_k) \quad (2)$$

where $X_1, X_2, X_3, \dots, X_n$: the input signal

$Wk_1, Wk_2, Wk_3, \dots, Wk_n$: the synaptic weight of the input signals

U_k: the output of the summing junction Σ

$\varphi(\cdot)$: the activation function

Multilayer Perceptrons(MLPs) are a very common type of ANN. They belong to feed-forward ANNs and can be trained using the supervised training method. Their training is based on the error back propagation algorithm, which was first formulated by Paul Werbos (Werbos, 1974). A number of variants of the back-propagation algorithm have been developed and widely used to train MLPs. These variants include back-propagation with momentum, Levenberg-Marquardt, Newton and Resilient back-propagation. The data used are divided into three categories: *training data*, *validation data* and *test data*. As its name indicates, the first type of data is used at the training stage to adapt the neurons' synaptic weights and bias.

The second type is used to monitor the training process and prevent overfitting, while the third type is not involved in the training, but only used to evaluate and compare different models. Upon designing and developing an ANN, a series of trials are performed, modifying various elements until the most appropriate ANN is developed to solve a specific problem. These modifications initially involve the network's architecture, as well as the training method and algorithm. They may also involve the number of the network's hidden layers or the number of hidden neurons in every hidden layer. Another modification involves the activation function used by each artificial neuron, as well as the participation of each of the three sets used.

Generally, the right selection of variants has a direct impact on the ANN's reliability. The various trials that are performed need to be followed by an evaluation process in order to select the best network. This evaluation process is based on the results of the test set. To this end, one or more criteria are selected from a range of criteria, and finally the best ANN is the one whose values for the selected criteria are the lowest in the test set. The main evaluation criteria which are used include the mean square error (MSE), root mean square error (RMSE), mean relative error (MRE), and mean absolute error (MAE) (Ananthraman&Garg, 1993).

3. Manipulator kinematics and dynamics

Kinematics is the science of motion which treats motion without regard to the forces which cause it. Kinematics of manipulators refers to all of the geometrical and time based properties of motion. Forward kinematics involves the computation of the position and orientation of the end-effector of the manipulator, given the joint angles. It maps the joint space coordinates to the Cartesian space coordinates and can be represented as $x = f(\theta)$ where θ is the vector of individual joint angles, and x is the end-effector location in Cartesian coordinates. The function f is a nonlinear function consisting of some trigonometric quantities. The mapping is relatively straightforward. Inverse kinematics involves the computation of all possible sets of joint angles which could be used to attain a desired position and orientation of the end-effector. Since the kinematic equations are nonlinear, the possibility of multiple solutions arises, and the resulting computations are much more intensive (Gullapalli *et al*., 1994).

Dynamics involves the study of forces required to cause motion and it is, like kinematics, a field of study in itself. In order for the robot to accelerate from rest, move at a constant velocity, and decelerate to a stop, a complex set of torque functions needs to be applied to the actuators at the joints. Dynamics involves the mapping between the joint torques and the resulting joint positions, velocities and accelerations and there are two problems involved. One, for controlling the manipulator to follow a desired path, the actuator torque functions can be calculated using dynamic equations of motion of the manipulator obtained from the Lagrange-Euler formulation or the Newton-Euler formulation. This computation involves inertia effects, Coriolis and centrifugal forces, friction, gravity and backlash in the case of gear driven robots (Bekey & Goldberg, 1993).

4. Manipulator control

Control of mechanical manipulators involves solving trigonometric relationships between the links of the manipulator and dynamic equations to produce particular motion of the end-effector. As more number of degrees of freedom and nonlinearities are introduced these computations become less and less tractable. The manipulation tasks performed by living organisms seems to suggest that solving trigonometricand dynamic equations is a very inefficient way of controlling the manipulator.Connectionist approaches to robot control are grounded in these observationsof biological systems. The emphasis is on learning the mapping between thevarious variables without an accurate knowledge of the system parameters or theequations governing the system. This is important from the point of view ofrobot control since many robot parameters are not known precisely.Given the above considerations it is intuitively very appealing to use neuralnetworks to solve the complex robot control problem by just learning from examplesof the robots behavior. The tremendous amount of work being reportedin application of neural networks to robot control attests to this notion. Theneural network approaches to robot motion control can be classified into threecategories: inverse kinematic control, dynamic control and sensor-based motioncontrol. Each of these three approaches is reviewed in the following sections (Sundareshan & Askew, 1994).

Robotics problem can be decomposed into the following three levels: task planning, path planning and motion control. Task planning involves receiving

instructions about the task to be performed and management and coordination of the information. Trajectory planning involves the determination of a sequence of points through which the robot end effector should pass subject to some constraints. Motion control involves the generation of the necessary joint torques to drive the robot to follow the desired trajectory. Artificial neural networks can be applied to all of the above areas. This paper will focus on neural network applications to the motion control problem involving both kinematic and dynamic issues.

4.1. Inverse kinematic control

Inverse kinematics refers to the problem of determining the robot joint angles to achieve a desired position and orientation of the robot end-effector. The Cartesian velocity of the tip of the arm can be related to the joint velocities by means of the Jacobian, J, as

$$\dot{x} = J(\theta)\dot{\theta} \tag{3}$$

where the elements of the Jacobian matrix J are the partial derivatives

The manipulator Jacobian is a function of the joint angles and as the manipulator moves, 0 changes and hence the Jacobian changes (Yu *et al*., 2014)

At any particular instant, however, the Jacobian performs a linear transformation. Jacobians are thus time-varying linear transformations. The joint velocities for a desired Cartesian velocity can be obtained by solving the above equation to yield:

$$\dot{\theta} = J^{-1}(\theta)\dot{x} \tag{4}$$

Thus, for a Cartesian space trajectory with velocity constraints, the desired motion of the joints can be calculated based on the above equation. This is referred to as velocity based inverse kinematic control or inverse Jacobian control. The scheme is shown in Fig. 2.

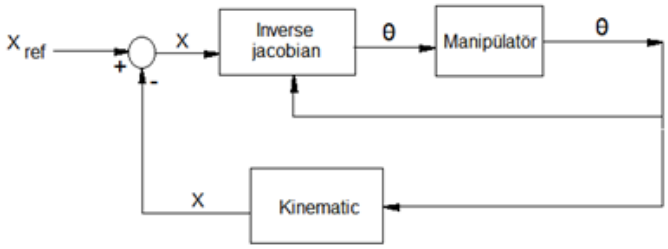

Figure 2. The control loop of invers Jacobian

If the Jacobian matrix is not invertible, as happens at singular points, the above method is not feasible. The method is critically dependent on efficient inversionof the Jacobian for successful application (Yegerlehner & Meckl, 1993). The inverse Jacobian for a simple two link manipulator with rotary joints, and link lengths l_1 and l_2 is given by

$$J^{-1}(\theta) = \frac{1}{l_1 l_2 s_2}\begin{bmatrix} l_2 c_{12} & l_2 s_{12} \\ -l_1 c_1 - l_2 c_{12} & -l_1 s_1 - l_2 s_{12} \end{bmatrix} \tag{5}$$

where s_i and c_i indicate the sine and cosine of angle θ_i, s_{ij} and c_{ij} indicate the sine and cosine of $(\theta i + \theta j)$ respectively. The expression becomes more complex as the degrees of freedom of the manipulator increase. Also, the robot kinematic parameters need to be accurately known for successful operation of this method. Neural networks do not need a priori knowledge of these parameters and thus offer an alternative to this method.

4.2. Robot dynamıcs control

Dynamic control of robots deals with the problem of computing the required torques for a desired motion. Since the motion of the links is coupled and the resuiting motion equations are highly nonlinear, the computations are considerably more complex than in the inverse kinematic case. The conventional way of controlling manipulators is by simple proportional integral derivative (PID) control, which is completely error driven. No attempt is made to decouple the motions of each joint and this causes errors which are suppressed by the error-driven control law. These controllers are based on the assumption of fixed robot parameters and are inadequate since the parameters of a robot depend on its configuration. When an accurate dynamic model of the mechanical manipulator is available, it may be used to calculate the joint drive torques for a desired trajectory (Santibañez *et al.*, 2010)

A controller is usually connected serially to the plant to be controlled. When a feedforward neural network is used as a controller the output of the network is the control input to the plant. Since the desired control action is unknown, it is not possible to determine the network error required to train the network by using the backpropagation algorithm. So suitable training schemes need to be developed for the use of neural networks for direct control.The neural network model is shown in Fig. 3. The model was applied to learning trajectory control of the robotic manipulator. The neural network modelwas implemented only the basal three degrees of freedom were controlled.

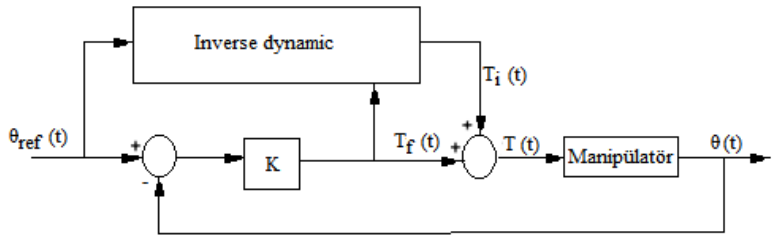

Figure 3. Manipulator control loop of inverse dynamic model

The total torque input to the manipulator $T(t)$ is a sum of the feedforward torque generated by the inverse-dynamics model $T_i(t)$, and the feedback torque $T_f(t)$. *The* desired trajectory is input to the model and the feedback torque is used as the error signal. Since the feedback torque is used as the error signal, the desire of the inverse dynamics of the manipulator is obtained by using the Lagrangian formulation. The structure is thus robot-dependent and the parameter values of the structure need to be estimated. Thus, this approach requires significant knowledge and needs to be implemented parallely to avoid the computation burden. However, the performance of the scheme is excellent. The model is learnt by repetitively experiencing a single

trajectory. As learning proceeds the inverse-dynamics model gradually takes the place of the feedback loop as the main controller. The trained network could control quite different and faster trajectories, and could also adapt to sudden changes in the dynamics of the manipulator. This overcomes the major drawback of the learning and adaptive control schemes discussed earlier: experiences obtained during learning could not be used for the execution of a quite different movement. Also the scheme requires considerably less memory than the table look-up approach (Liu & Wang, 2011).

The control system is set up as schematically illustrated in Figure 4.

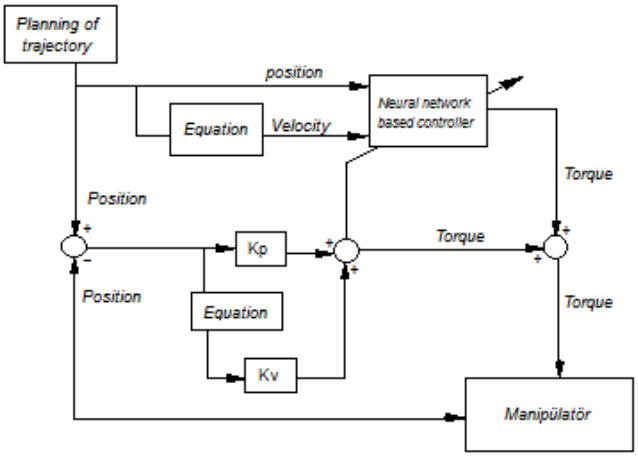

Figure 4. Neural network based controller block diagram

A trajectory planner is used to generate desired values of position, velocity and acceleration of the robot joints, for a prespecified trajectory. The neural network comprised the controller that represented pockets of the inverse dynamics of the system. It generates the actuator torques at the joints corresponding to the values of the input trajectory vector variables. The neural controller is used in tandem with a linear proportional derivative (PD) feedback controller. The torque input to the robot system is the sum of the torques generated by the feedforward neural controller and the feedback linear controller (Zhu & Zhang, 2011).

5. Conclusion

Industrial robots are man-made and hence the designer has a significant amount of knowledge of the system, and this knowledge must be used for control of the system rather than using a neural network and starting from zero knowledge as happens with the random initialization of weights. Neural networks are essentially function approximators. The degree of accuracy of the approximation is very critical if neural networks are to be used in real world situations. Thus neural network approaches should be compared with conventional approaches to determine their real strengths and weaknesses.The variety of neural networks applications to robot motion-control problems reviewed in this paper highlight the increasing importance of neural networks in the development of intelligent systems.

The work done so far can be considered to be experimental in nature and lacks a firm theoretical foundation. There are no set guidelines as to the type of network architecture, number of layers, the necessary number of units per layer, the nonlinear transfer function to be used and the learning rate suitable for a particular problem. This

can partly be explained due to the relative youth of the field, but efforts are needed to develop a theoretical structure for the field. Recently, Fuzzy Logic Control (FLC) based on Fuzzy Set Theory is being used in the control of ill-defined and complex systems. The fusion of neural networks and fuzzy logic would help complement the advantages that each of them possess individually: learning can be incorporated infuzzy logic and a systematic structure can be incorporated in neural networks.

The hybrid approaches discussed in this section attempt to overcome the weaknesses of each individual decision making structure by combining it with another complimentary decision making structure. It is the authors' firm belief that these kinds of hybrid, hierarchical systems hold the key to intelligent control systems.

References

Abdelhameed, M.M. (2005). Enhancement of sliding mode controller by fuzzy logic with application to robotic manipulators. *Mechatronics*, 15, 439–458.

Bekey, G.A. & Goldberg, K.Y. (1993). *Neural Networks in Robotics.* Kluwer Academic Publishers, Norwell, MA.

Cirak B. & Demirtas S. (2014). *An application of artificial neuralnetwork for predicting engine torque in a biodiesel engine.* American Journal of Energy Research, *1*(2), August.

Gullapalli, V., Barto, A.G. & Grnpen, R.A. (1994). Learning admittance mappings for force-guided assembly. In *Proc. IEEE Int. Conf. Robot. Automat.*, 2633-2638.

Katic, D. & Vukobratovic, M. (1994). Connectionist approaches to the control of manipulation robots at the executive hierarchical level: An overview. J. *Intelligent Robot. Systems* 10, 1-36.

Liu, J. & Wang, X. (2011). *Sliding Mode Control for Robot*. Springer, Berlin.

Liu, S. & Asada, H. (1993). Task-level on-line learning using neural networks and it application to deburring robots. *Advances Robot., Mechatronics and Haptic Interfaces* DSC-49, 333-338.

Prabhu, S.M., & Garg, D.P. (1996). Artificial neural network based robot control: An overview. *Journal of Intelligent and Robotic Systems*, *15*(4), 333-365.

Santibañez, V., Camarillo, K., Javier, M.V. & Campa, R. (2010). A practical PID regulator with bounded torques for robot manipulators. *Int. J. Control Autom. Syst*., *8*(3), 544-555.

Yegerlehner, J.D. & Meckl, E.H. (1993). Experimental evaluation of neural network controller for robot undergoing large payload changes. In *Proc. IEEE Int. Conf. Robot. Automat.*, 744-749.

Yu, W.S., Karkoub, M., Wu, T.S.& Her, M.G. (2014). Delayed output feedback control for nonlinear systems with two-layer interval fuzzy observers. *IEEE Trans. Fuzzy Syst*., 22(3), 611-630

Zhu, C. & Zhang, H. (2017). *Parallel Robotics Control Strategy Study Based on Fuzzy-PID*, 664–669. *Springer*, Berlin.