NXT* SCARA Model Based Design Controlled by Neural Network

A. A. H. Sallam¹, W. M. F. Abouzaid²,*

¹Computers and Control Engineering Department, Faculty of Engineering, Tanta University, Tanta, Egypt
²Electrical Engineering Department, Faculty of Engineering, Kafrelsheikh University, Egypt

Abstract This paper describes experimental results applying Artificial Neural Networks (ANNs) to perform the position control of a real SCARA manipulator robot. This approach has performed very successfully, with better results obtained with the Radial Basis Function (RBF) networks when compared to P controller and sliding mode positional controller. For multi-input multi-output (MIMO) continuous-time nonlinear systems, there are a few results available due to the difficulty in handling the coupling matrix between different inputs. A stable neural network controller was developed for a class of nonlinear multi-variable systems. The nonlinearities unknowns in the systems or in the controllers are approximated by linearly or nonlinearly parameterized neural networks, such as Radial Basis Function Neural Networks (RBF NNs) and Multilayer Neural Networks (MNNs). The introduction is talking about ANN which can be learned by many methods to control non linear system SCARA and determine the effective computational technique structure simple as possible give high efficiency compared with classical P-controller. The NXT SCARA modeling with two link planar robot arm, its kinematics and inverse kinematics with link motion equations, explanation for trajectory making of the edge of the robot arm. Then describes the controller design, its inputs, outputs and how the system tracking NN controller using RBFNN that is attractive for many problems, which give rapid settling time, no overshoot and reduce error. The simulation and results are explained which present the proposed NN improves the response and realizes good dynamic tracking and robustness of system nonlinearity with self tuning NN without change the system parameters. The results are calculated by MATLAB/SIMULINK.

Keywords Non-linear MIMO Systems, Artificial Neural Network, SCARA Model, RBFNN

1. Introduction

Robotics has become recently an interesting area of research, especially the robot manipulator control. Industrial robot manipulators of high accuracy require complicated methods of control, which is a nonlinear system with high coupling terms whose dynamics consists of uncertainty and has encountered with payload changes, friction and disturbance[1]. Applying a control technique is important to guarantee high efficiency and lower error for the robot's motion.

ANNs have been successfully applied to a number of scientific and engineering fields in recent years, such as function approximation, system identification and control, signal and image processing, time series prediction and so on. The major problems in designing of ANNs for a given problem or how to design a satisfactory ANNs’ architecture and which kind of learning algorithms can be effectively used for the training of ANNs. We have got weights and biases of ANNs which can be learned by many methods, i.e., back-propagation algorithm, genetic algorithm, evolutionary programming, random search algorithm and so on. Usually, a neural network’s performance is highly dependent on its structure[2]. The main goal of this work is to explore an effective computational intelligence technique to control this complex system using a structure as simple as possible, with preference to one that requires less change in the previous conventional controller that has installed in the system, do not overload the main processor and is robustness against disturbance and load effect variations[3].

ANNs have been proven to be universal approximates of non-linear dynamic systems. They are able to emulate any complex nonlinear dynamic system by using an appropriate MNN. After being used for many years in pattern recognition signal and image processing applications, ANNs are now employed in a larger class of scientific disciplines[4].

It has been pointed out that multi-layered neural network can be used for the approximation of any nonlinear function. They make such artificial intelligence technology attractive not only in the application areas such as pattern recognition, information and graphics processing, but also in the
intelligent control of nonlinear and complicated systems such as robot manipulators. A new field in robot control using neural network technology is beginning to emerge to deal with the issues related to the dynamics in the robot control design. A neural network based dynamics compensation method has been proposed for trajectory control of a robot system[5].

Nonlinear control systems are those control systems where nonlinearity plays a significant role, either in the controlled process (plant) or in the controller itself. Linear plants arise naturally in numerous engineering and natural systems, including mechanical and biological systems, aerospace and automotive control, industrial process control, and many others. Nonlinear control theory is concerned with the analysis and design of nonlinear control systems. It is closely related to nonlinear systems theory in general, which provides its basic analysis tools[6]. For most practical nonlinear physical systems, it is often very difficult, almost two; to describe them by sufficiently simplified analytical models. In view of these, there is great interest in developing a robust and less model-dependent methodology for representing a complex nonlinear dynamic system. To avoid the difficulties experienced in the classical nonlinear system modeling, neural-network-based nonlinear system modeling methods appeared to be an appealing alternative. The rationale behind this approach lies in the fact that a multilayer neural network.

Figure 1. With an appropriate nonlinear activation function can approximate any nonlinear relationship. Many types of neural networks have been developed for tackling different problems, but two types have received the most attention in recent years, the 1st type is multilayer feed forward neural networks and recurrent networks. And the 2nd type is Multilayer feed forward networks have proved extremely successful in pattern recognition problems, and recurrent networks for dynamical system modeling and time-series forecasting[7].

NXT SCARA is two-link planar robot arm built with LEGO Mindstorms NXT SCARA stands for Selective Compliance Assembly Robot Arm. These type robotic manipulators have been used extensively for industrial automation, and are capable of performing a wide variety of precise pick and place operations in a variety of industrial applications[8]. When using NN as a controller, without using Jacobian of an objective plant it is possible to apply the learning scheme directly. Those neuro-controller learns the inverse dynamics of the SCARA robot.[9].

2. NXT SCARA Modeling

This section describes a geometric model of NXT SCARA and inverse kinematics that determines each link angles in order to achieve a desired pose / position.

In general, SCARA has four degrees of freedom whereby two or three horizontal servo controlled joints are shoulder, elbow, and wrist while the forth movement is actuated pneumatically. The SCARA robot is known for its ability to perform high-speed operations. A particular feature is its selective adoption that is extremely helpful in assembling operations that require the insertion of objects in pallets. Therefore, SCARA, a horizontal evaluate configuration robot has 4 degrees of freedom in which two or three horizontal servo controlled joints are the shoulder, elbow and wrist while the last vertical axis is pneumatically controlled. Several different tasks can be performed by a SCARA such as pick and place, welding, painting, brushing, and peg-in-hole[10]. SCARA designed in Japan, I generally suited for small parts insertion tasks for assembly lines like electronic component insertion. Although the final aim is real robotics, it is often very useful to perform simulations[11].

2.1. Two – Link Planar Robot Arm

NXT SCARA can be considered as a two-link planar robot arm with a coordinate system shown in Figure 2.

Since \( \theta_{l,2} : \text{link angle} \) and \( \theta_{m,2} : \text{DC motor angle} \)

Physical parameters of NXT SCARA are the following

2.2. Kinematics and Inverse Kinematics

Kinematics is a part of mechanics studying motion without considering the forces which are responsible for this motion. A motion is in general described by trajectories, velocities and accelerations. In the robot joints, the trajectories are measured either as the angle in a rotational joint or as the distance in a translational joint. The joint...
variables are also called internal coordinates. When planning and programming a robot task the trajectory of the robot endpoint is of importance. There are two kinds of kinematics; the 1st one is direct kinematics and the 2nd is inverse kinematics. It should be noted that a fixed mechanical stiffness at the joints allows only a fixed Cartesian compliance for a given robot configuration. Modification of this compliance requires the presence of kinematic redundancy (changing the robot configuration at a given robot end-effector pose) and/or the tuning of the positional gains in a feedback control law[12]. Direct kinematics in the case of a two-segment robot represents the calculation of the position of the robot endpoint from the known joint angles. Inverse kinematics calculate the joint variables from the known position of the robot endpoint[13]. Therefore, inverse kinematics are important to drive the unknown link motions from the known edge motion.

\[
\mathbf{p}_1 = l_1 \begin{bmatrix} \cos \vartheta_1 \\ \sin \vartheta_1 \end{bmatrix}
\]

where vector \( \mathbf{p}_1 \) is directed along the first segment of the simple mechanism

\[
\mathbf{p}_2 = l_2 \begin{bmatrix} \cos(\vartheta_1 + \vartheta_2) \\ \sin(\vartheta_1 + \vartheta_2) \end{bmatrix}
\]

where vector \( \mathbf{p}_2 \) is along with the second segment.

\[
x = \begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} l_1 \cos \vartheta_1 + l_2 \cos(\vartheta_1 + \vartheta_2) \\ l_1 \sin \vartheta_1 + l_2 \sin(\vartheta_1 + \vartheta_2) \end{bmatrix}
\]

And vector \( \mathbf{x} \) describes the position of the robot endpoint.

When solving the inverse kinematics, we calculate the joint angles from the known position of the robot endpoint. The cosine rule is used

\[
x^2 + y^2 = l_1^2 + l_2^2 - 2l_1l_2 \cos(180^\circ - \vartheta_2)
\]

The angle in the second joint of the two-segment manipulator is calculated as the inverse trigonometric function

\[
\vartheta_2 = \arccos \frac{x^2 + y^2 - l_1^2 - l_2^2}{2l_1l_2}
\]

Again, we use of the inverse trigonometric functions

\[
\vartheta_1 = \arctan \left( \frac{y}{x} \right) - \arctan \left( \frac{l_2 \sin \vartheta_2}{l_1 + l_2 \cos \vartheta_2} \right)
\]

As shown, we have two solutions, “elbow-up” and “elbow-down”, in Figure 3, where there are two poses at the same position of the edge.

2.3. Trajectory Making

To make the reference trajectory of the edge of the robot arm, we have to describe a trajectory as a time – varying function. We often select straight line or circular arc as a basic trajectory.

- straight line : \( \eta(t) \) means the length of the line
- circular arc : \( \eta(t) \) means the angle of the circle

To calculate the position \( x(t), y(t) \) corresponding to basic trajectory \( \eta(t) \), and convert them to \( \vartheta_1(t), \vartheta_2(t) \) with inverse kinematics. The making procedure of a reference trajectory by using basic trajectory shown in Figure 4.

2.4. CP motion and PTP Motion

There are two motions of a robot arm called as CP (Continuous Path) motion and PTP (Pose To Pose) motion. In the case of NXT SCARA, CP motion corresponds to draw with a pen, and PTP motion does to move a pen without drawing.

3. NXT SCARA Controller Design

In the following point, describe NXT SCARA controller design for tracking reference trajectory.

![Figure 4. Making procedure of a reference trajectory with basic trajectories](image)
Motion Planning $\rightarrow$ Trajectory Parameter $\rightarrow$ Trajectory Making $\rightarrow$ Trajectory (Reference) $\rightarrow$ Controller $\rightarrow$ Control Input $\rightarrow$ Plant $\rightarrow$ Feedback

**Figure 5.** Motion control system

Position Reference $\rightarrow$ Inverse kinematics $\rightarrow$ Backlash 1 Compensation $\rightarrow$ Neural Network $\rightarrow$ PWM1 saturation

NXT SCARA $\rightarrow$ $\theta_{1n}$ $\rightarrow$ $\theta_{2n}$ $\rightarrow$ Backlash 2 Compensation $\rightarrow$ Neural Network $\rightarrow$ PWM2 saturation

**Figure 6.** NXT SCARA tracking controller block diagram

$\phi_1$ $\rightarrow$ $\phi_2$ $\rightarrow$ $\phi_n$ $\rightarrow$ $u_1(t)$ $\rightarrow$ $u_2(t)$ $\rightarrow$ $u_M(t)$ $\rightarrow$ $y_1(t)$ $\rightarrow$ $y_2(t)$ $\rightarrow$ $y_p(t)$

**Figure 7.** A MIMO RBFNN
3.1. Inputs & Outputs

The inputs to the actuators are PWM duty of DC motors and the outputs from the sensors are rotation angles of the motor. The rotation angle of the links can be derived from dividing motor angle by the gear reduction ratio.

3.2. Backlash

The driving gear trains have some backlash. It has a negative impact on tracking accuracy because it makes some lost motion when movement is reversed and contact is re-established. We need to compensate it for engaging the gears.

The proposed neural network self tuning control structure uses a separate network for each link. This improves the obtained trajectory because any noise or disturbance if it is subjected to one link will not affect on the other[1]. A block diagram of the NXT SCARA tracking controller shown in Figure 6. There is really calculations for restricting link angles.

NN can be employed in advanced intelligent control applications by making use of their non linearity learning, parallel processing and generalization capacities[4]. A Neural Network is constituted of densely interconnected neurons. A neuron is a computing node. It performs the multiplication of its inputs by constant weights, sums the results, shifts it by a constant bias and maps it to a nonlinear activation function before transferring it to its output. A feed forward neural network is organized in layers of neurons: an input layer, one or more hidden layers and an output layer. The inputs to each neuron of the input layer are the inputs to the network. The inputs to each neuron of the hidden or output layer are the outputs from the neurons of the preceding layer. For a given modeling problem, the numbers of nodes in the input and output layers are determined from the physics of the problem, and equal to the numbers of input and output parameters respectively, while the number of nodes in the hidden layers(s) is determined by trial and error[14]. Figure 7. represents Radial Basis Functions Neural Networks (RBFNN) that is a type of feed forward neural network which learning involves only one layer with lesser computations. These features make RBFNN attractive in many practical problems[15].

4. Simulation and Results

All simulations were presented using MATLAB and SIMULINK, which are used widely in control applications.

Simulation results show that the designed neural controller realizes a good dynamic behavior of the motor, with a rapid settling time, no overshoot, almost instantaneous rejection of load disturbance, a perfect speed tracking and it deals well with parameter variations of the motor. It seems to be a high-performance robust controller. Then two – link robot arm is considered to follow a MATLAB logo trajectory. The actual trajectories for link1 and link2 are shown in Figure 8. and Figure 9, respectively.

At the beginning of the simulation, the error is maximized because the NN has not been learned yet. But after 3ms of time, the error is reduced quickly as shown in Figure10, and the robot starts to follow the desired trajectory efficiently. The proposed NN self tuning improves the response through reducing the settling time, overshoot and steady state error.
Figure 11. and Figure 12, represent the simulation response of theta1 and theta2 (output of the system) for P controller and NN.

Figure 11. Simulated response of theta1 for P controller and NN for MATLAB logo trajectory.

Figure 12. Simulated response of theta2 for P controller and NN for MATLAB logo trajectory.
- The controller block runs in discrete – time (base sample time=1ms) and the plant (NXT SCARA subsystem) runs in continuous – time (sample time=0ms).
- The simulation stops when both PWM1 and PWM2 equal to zero.
- We compared the results of P – controller classical tuning method and NN self tuning method in terms of overshoot, transient response, and steady state error.

Then, Figure 13., and Figure 14., represents the output of controller PWM1 and PWM2 for both controllers P controller and NN respectively.

Figure 13. Output of PWM1 of P controller and NN for MATLAB logo trajectory

Figure 14. Output of PWM2 for P controller and NN for MATLAB logo trajectory
4.1. Backlash Compensation

The backlash compensation starts when a sign of reference motor angle variation is reversed. In Figure 15, and Figure 16, we can see an effect of backlash compensation of the output of controller PWM1 for NN and P controller for MATLAB logo trajectory.

![Figure 15](image1.png)  
*Figure 15. PWM1 for NN for MATLAB logo trajectory*

![Figure 16](image2.png)  
*Figure 16. PWM1 for P controller for MATLAB logo trajectory*
5. Conclusions

This paper has presented a practical and successful application of a neural controller performing in parallel with a conventional controller in the position and trajectory following control of a real robot. Simulation studies show that proposed NN self-tuning provides excellent tracking and robustness in the presence of system nonlinearity, since the NN is trained on-line, therefore any changes in system parameters will cause the NN to provide the appropriate change in the control signal to resume the effect. The main advantage of this architecture is that it does not require any modification of the previous conventional controller algorithm. The use of a conventional controller performing in parallel with the NNs is advantageous to maintain the robustness of the system when the NN become saturated (due to high learning rates) and it is important to force the readjustment of the synaptic weights of the NN used when the robot changes its configuration.

REFERENCES


