Neural Network Based Inverse Kinematics Solution for 6-R Robot Using Levenberg-Marquardt Algorithm

Prashant Badoni

Mechanical Engineering Department, Graphic Era University, Dehradun - 248002, India

Abstract: The traditional approaches are insufficient to solve the complex kinematics problems of the redundant robotic manipulators. To overcome such intricacy, ANNs are used nowadays. The performance of the neural network is affected by the training algorithm and network topology. There are numerous training algorithms which are used in the training of neural networks. In this paper, Levenberg-Marquardt is used in training algorithm and its effect on the performance of the neural network on the inverse kinematics model learning of a 6-R robot is studied.

Keywords: Inverse Kinematics Solution, MATLAB Toolbox, Neural Networks, Robot manipulator, Training Algorithm.

1. Introduction

Neural network is one of the prominent artificial intelligent techniques used in the robotics to accomplish more intelligence in systems with high degree of autonomy. ANN incorporates learning ability which provides flexibility to the robotic systems. Neural network can be implemented using MATLAB software. Procedure to train the neural network model is as follows:

- Data collection
- Network creation
- Network configuration
- Initialization of the weights and biases
- Network training
- Network validation
- Use the network

The working principal of a neural network is based on learning from the formerly obtained data set known as training set, and then go through the success of system using test data. The learning algorithm affects the employment of the neural network greatly. In this study, the effects of Levenberg-Marquardt learning algorithm have been tested for the inverse kinematics solution of a six joint robotic manipulator.

This paper is organized as follows: Section II provides the kinematics analysis of the 6-R robot. Section III of the paper deals with the neural network based inverse kinematics solution. Section IV describes training and testing. Section V gives results and a discussion, and finally Section VI concludes the paper.

2. Kinematic Analysis of 6-R Robot

A Robot manipulator is composed of a group of links (rigid bodies) connected together by revolute or prismatic joints which allow motion for the desired link. Robot Kinematics refers to the analytical study of the motion of a robot manipulator without regard to any factor (like force) which influence the robot movement. Robot Kinematics can be split into forward and inverse kinematics. In the forward kinematics problem, the end effector’s location in the work space, that is position and orientation, is determined based on the joint variables [1] [2] [3]. The forward kinematics problem may express mathematically as follows:

\[ F(\theta_1, \theta_2, \theta_3, ..., \theta_n) = [p_x, p_y, p_z, R] \]

Where, \( \theta_1, \theta_2, \theta_3, ..., \theta_n \) are the input variables, \([p_x, p_y, p_z]\) are desired position and \( R \) is the desired rotation.

The inverse kinematics problem refers to finding the values of the joint variables that allows the manipulator to reach the given location. The inverse kinematics problem can be expressed mathematically as follows:

\[ F(p_x, p_y, p_z, R) = (\theta_1, \theta_2, \theta_3, ..., \theta_n) \]

The joint variables are the link extension in the case of prismatic joints, or the angles between the links in case of rotational joints.

Figure 1: D-H coordinates of the robot

![Figure 1: D-H coordinates of the robot](image)

Figure 1 depicts the structure and coordinates of 6-DOF robot manipulator which is studied in during the work.

The D-H parameters of the manipulator are listed in Table 1.

<table>
<thead>
<tr>
<th>Joints</th>
<th>( a_i )</th>
<th>( \alpha_i )</th>
<th>( d_i )</th>
<th>( \theta_i )</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>( \theta_1 )</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>( \pi/4 )</td>
<td>0</td>
<td>( \theta_2 )</td>
</tr>
<tr>
<td>3</td>
<td>( l_2 )</td>
<td>( \pi/4 )</td>
<td>( l_1 )</td>
<td>( \theta_3 )</td>
</tr>
</tbody>
</table>
According the Denavit-Hartenberg method, the transformation can be formulated in the chain product of six successive homogeneous matrices $^{i-1}T_i$. 

$$i-1 \frac{1}{6} T$$ is the homogeneous transformation matrix relating the $i^{th}$ coordinate frame to the $(i-1)^{th}$ coordinate frame [4].

$$\frac{1}{6} T = \frac{1}{6} T_{1,2} \ldots \frac{1}{6} T_{n-1,n} (1)$$

Equation (1) contains a large set of trigonometric functions:

$$\sin \theta_i = S_i, \text{ and } \cos \theta_i = C_i$$

$\frac{1}{6} T$ can be calculated by the following:

$$\frac{1}{6} T = \begin{bmatrix} n_x & o_x & a_x & p_x \\ n_y & o_y & a_y & p_y \\ n_z & o_z & a_z & p_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$ (2)

In (2), $n_x$, $n_y$, $n_z$, $o_x$, $o_y$, $o_z$, $a_x$, $a_y$, $a_z$, show the rotational elements of the transformation matrix and $p_x$, $p_y$, $p_z$ refer to the elements of the position vector.

$$n_x = -C_1 C_2 C_3 S_6 + S_1 S_3 C_6 - C_1 C_3 S_4 S_6 - C_1 S_3 S_4 C_6 - S_1 C_3 C_6$$

$$n_y = -S_1 S_2 C_3 S_6 - C_1 S_3 C_4 S_6 - S_1 C_2 C_4 S_6 - C_2 S_3 S_4 C_6 + C_1 C_3 C_6$$

$$n_z = C_2 C_3 C_4 S_6 - S_3 S_4 S_6 + C_1 S_3 C_6$$

$$o_x = -C_1 S_2 C_3 S_6 + S_1 S_3 C_6 - C_1 C_2 S_4 S_6 + C_2 S_3 S_4 C_6 + C_1 S_3 C_6$$

$$o_y = -S_1 S_2 C_4 S_6 - C_1 S_3 C_4 S_6 - S_1 C_2 S_4 S_6 + S_1 C_3 C_6 + S_3 S_4 C_6$$

$$o_z = C_2 C_3 C_4 S_6 - S_3 S_4 S_6 - C_1 S_3 C_6$$

$$a_x = -C_1 S_4 C_3 S_6 + S_1 S_3 C_6 + C_2 S_3 S_4 C_6 + S_3 S_4 C_6$$

$$a_y = -S_1 S_2 C_4 S_6 - C_1 S_3 C_4 S_6 - S_1 C_2 S_4 S_6 + S_1 C_3 C_6 + S_3 S_4 C_6$$

$$a_z = C_2 C_3 S_4 S_6 + S_3 S_4 C_6$$

$$p_x = (C_1 C_2 S_4 S_6 + S_1 S_3 C_6 + C_1 C_2 C_3) I_3 + (S_1 C_3 S_4 S_6 + S_1 S_3 S_4 C_6 + C_1 C_2 C_3) I_1$$

$$p_y = (S_1 S_2 C_4 S_6 + C_1 S_3 C_4 S_6 + S_1 C_2 S_4 S_6 + S_1 C_3 C_6 + S_3 S_4 C_6) I_3 + (S_1 S_2 S_4 C_6 + C_2 S_3 S_4 C_6 + S_1 C_2 C_3) I_1$$

$$p_z = (C_2 C_3 S_4 S_6 + S_3 S_4 C_6) I_3 + (C_2 C_3 S_4 S_6 + S_3 S_4 C_6) I_1 + I_2 I_3 + I_1$$

The inverse kinematics solution for the robot is indicated as follows:

$$\cos \theta_i = -L_{i}^{2} + L_{i}^{2} - d_{i}^{2} / 2 L_{i} L_{i} I_{i} (3)$$

The equation obtained from (2) as:

$$(S_{15} I_{5} + S_{4} I_{4})^{2} + (C_{45} I_{5} + C_{4} I_{4} + I_{3}^{2})^{2} = p_{x}^{2} + p_{y}^{2} + (p_{x} - I_{1})^{2} (4)$$

Then,

$$\theta_3 = \arcsin \left[ \left( p_{x}^{2} + p_{y}^{2} + (p_{x} - I_{1})^{2} \right) / A \right] - \arctan \left[ \left( C_{4} I_{4} + I_{3}^{2} - (C_{4} I_{4} + I_{3}^{2}) \right) / A \right] \left( 5 \right)$$

We could also obtain the following expressions:

$$\theta_2 = \arcsin \left[ \left( (C_{45} I_{5} + C_{4} I_{4} + I_{3}^{2}) / \sqrt{ (p_{x}^{2} + p_{y}^{2} + (p_{x} - I_{1})^{2}) } \right) - \arctan \left[ p_{x} / (p_{y} - I_{1}) \right] \right]$$

$$\theta_1 = \arcsin \left[ \left( (C_{2} C_{3} S_{6} - C_{2} C_{4} - C_{2} C_{6}) / \sqrt{ (p_{x}^{2} - a_{3} I_{3})^{2} + (p_{y} - a_{3} I_{3})^{2} } \right) - \arctan \left[ (p_{x} - a_{3} I_{3}) / (p_{y} - a_{3} I_{3}) \right] \right]$$

The strategy to solve inverse kinematics problem tend to be time consuming, so there is usually low interest in applying this technique for kinematic calculations. The trained neural network can give the inverse kinematics solution quickly for any given Cartesian coordinate in a robotic system.

### 3. Neural Network Based Inverse Kinematics Solution

Neural networks are generally used in the modeling of nonlinear processes. ANN is a parallel-distributed information processing system. To form a trainable nonlinear system, it stores the samples with distributed coding. Training of a neural network can be expressed as a mapping between any given input and output data set. Neural networks have some advantages, such as adoption, learning and generalization. Implementation of a neural-network model requires us to decide the structure of the model, the type of activation function and the learning algorithm.

In Figure 3, the schematic representation of a neural network based inverse kinematics solution is given. The solution system is based on training a neural network to solve an inverse kinematics problem based on the prepared training data set using direct kinematics equations. In Figure 2, “e” refers to error – the neural network results will be an approximation, and there will be an acceptable error in the solution.
Figure 2: ANN based inverse kinematics solution system [5]

The designed neural-network topology is given in Figure 3. A feed-forward multilayer neural-network structure was designed including 12 inputs and 6 outputs. Only one hidden layer was used during the study.

![Neural Network Diagram](image)

Figure 3: Structure of neural network used in this study

4. Training Method

To train the network we must provide the ANN with the dataset. ANN is trained with the data which is generated by a fifth-order polynomial trajectory planning algorithm. The equation for fifth-order polynomial trajectory planning is given in following equation:

\[
\theta_i(t) = \theta_{io} + 10/t_i^5(\theta_{if} - \theta_{io}) t^5 + 15/t_i^4(\theta_{if} - \theta_{io}) t^4 + 6/t_i^3(\theta_{if} - \theta_{io}) t^3; \quad i = 1, 2, 3 \ldots \quad n \quad (6)
\]

Where,

- \( \theta_i(t) \) = angular position at time t
- \( \theta_{io} \) = initial position of the \( i^{th} \) joint
- \( \theta_{if} \) = final position of the \( i^{th} \) joint
- \( n \) = number of joints
- \( t_i \) = arrival time from initial position to the target

Initial and final angular positions are defined to produce data in the workspace of robot. After gathering data from the whole network is trained in the back propagation mode and all the weights are updated according to the new training data. For the training, 100 data values corresponding to the \((\theta_1, \theta_2, \theta_3 \ldots \theta_n)\) joint angles according to the different \((n, o_x, a_x, p_x, n_y, o_y, a_y, p_y)\) Cartesian coordinate parameters were generated by using (6) based on kinematic equations given in (2). A sample data set produced for the training of neural networks is given in Table 2 and 3.

### Table 2: A sample input data set for the training of neural networks

<table>
<thead>
<tr>
<th>Inputs</th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>( n_x )</td>
<td>( n_y )</td>
<td>( n_z )</td>
<td>( o_x )</td>
<td>( o_y )</td>
<td>( o_z )</td>
<td>( a_x )</td>
<td>( a_y )</td>
<td>( a_z )</td>
<td>( p_x )</td>
</tr>
<tr>
<td>-0.9467</td>
<td>-0.0186</td>
<td>-0.3216</td>
<td>-0.3188</td>
<td>-0.0909</td>
<td>0.9435</td>
<td>-0.0468</td>
<td>0.9957</td>
<td>0.0802</td>
<td>10.7388</td>
</tr>
<tr>
<td>0.4031</td>
<td>-0.4414</td>
<td>-0.8017</td>
<td>-0.8689</td>
<td>-0.4595</td>
<td>-0.1839</td>
<td>-0.2872</td>
<td>0.7707</td>
<td>-0.5688</td>
<td>-15.6523</td>
</tr>
<tr>
<td>-0.9209</td>
<td>0.3787</td>
<td>-0.0921</td>
<td>-0.3898</td>
<td>-0.8943</td>
<td>0.2199</td>
<td>0.0009</td>
<td>0.2384</td>
<td>0.9712</td>
<td>1.0641</td>
</tr>
<tr>
<td>0.7646</td>
<td>-0.3951</td>
<td>-0.5091</td>
<td>0.6249</td>
<td>0.6478</td>
<td>0.4358</td>
<td>0.1576</td>
<td>-0.6514</td>
<td>0.7422</td>
<td>-9.4243</td>
</tr>
<tr>
<td>0.2783</td>
<td>0.4263</td>
<td>-0.8607</td>
<td>0.9056</td>
<td>-0.4151</td>
<td>0.0872</td>
<td>-0.32</td>
<td>-0.8037</td>
<td>-0.5016</td>
<td>5.307</td>
</tr>
<tr>
<td>0.1042</td>
<td>-0.5075</td>
<td>-0.8553</td>
<td>0.0489</td>
<td>0.8616</td>
<td>-0.5053</td>
<td>0.9934</td>
<td>0.0108</td>
<td>0.1146</td>
<td>-1.071</td>
</tr>
<tr>
<td>-0.7806</td>
<td>0.5435</td>
<td>0.3086</td>
<td>0.4541</td>
<td>0.1539</td>
<td>0.8775</td>
<td>0.4295</td>
<td>0.8251</td>
<td>-0.367</td>
<td>-2.8101</td>
</tr>
<tr>
<td>-0.4807</td>
<td>0.2339</td>
<td>-0.8451</td>
<td>0.6899</td>
<td>0.6958</td>
<td>-0.1999</td>
<td>0.5413</td>
<td>-0.6791</td>
<td>-0.4958</td>
<td>2.7343</td>
</tr>
<tr>
<td>-0.7663</td>
<td>-0.6233</td>
<td>-0.1561</td>
<td>-0.4972</td>
<td>0.4213</td>
<td>0.7585</td>
<td>-0.407</td>
<td>0.6588</td>
<td>-0.6327</td>
<td>-8.1384</td>
</tr>
<tr>
<td>-0.4693</td>
<td>0.873</td>
<td>-0.1325</td>
<td>0.8665</td>
<td>0.4264</td>
<td>-0.2594</td>
<td>-0.1699</td>
<td>-0.2366</td>
<td>-0.9566</td>
<td>-7.5946</td>
</tr>
<tr>
<td>-0.5937</td>
<td>-0.5062</td>
<td>-0.6255</td>
<td>-0.7906</td>
<td>0.5119</td>
<td>0.3361</td>
<td>0.1501</td>
<td>0.694</td>
<td>-0.7041</td>
<td>-4.8593</td>
</tr>
</tbody>
</table>
Table 3: A sample target data set for the training of neural networks

<table>
<thead>
<tr>
<th>Targets</th>
<th>θ₁</th>
<th>θ₂</th>
<th>θ₃</th>
<th>θ₄</th>
<th>θ₅</th>
<th>θ₆</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>169.779</td>
<td>74.3217</td>
<td>-79.0853</td>
<td>169.779</td>
<td>-6.3597</td>
<td>169.779</td>
</tr>
<tr>
<td></td>
<td>176.5911</td>
<td>78.1062</td>
<td>-73.0301</td>
<td>176.5911</td>
<td>-2.1211</td>
<td>176.5911</td>
</tr>
<tr>
<td></td>
<td>183.4089</td>
<td>81.8938</td>
<td>-66.9699</td>
<td>183.4089</td>
<td>2.1211</td>
<td>183.4089</td>
</tr>
<tr>
<td></td>
<td>190.221</td>
<td>85.6783</td>
<td>-60.9147</td>
<td>190.221</td>
<td>6.3597</td>
<td>190.221</td>
</tr>
<tr>
<td></td>
<td>197.0165</td>
<td>89.4536</td>
<td>-54.8742</td>
<td>197.0165</td>
<td>10.588</td>
<td>197.0165</td>
</tr>
<tr>
<td></td>
<td>203.7842</td>
<td>93.2135</td>
<td>-48.8585</td>
<td>203.7842</td>
<td>14.7991</td>
<td>203.7842</td>
</tr>
<tr>
<td></td>
<td>210.5132</td>
<td>96.9518</td>
<td>-42.8772</td>
<td>210.5132</td>
<td>18.986</td>
<td>210.5132</td>
</tr>
<tr>
<td></td>
<td>217.1925</td>
<td>100.6625</td>
<td>-36.94</td>
<td>217.1925</td>
<td>23.142</td>
<td>217.1925</td>
</tr>
<tr>
<td></td>
<td>223.8114</td>
<td>104.3396</td>
<td>-31.0566</td>
<td>223.8114</td>
<td>27.2604</td>
<td>223.8114</td>
</tr>
<tr>
<td></td>
<td>236.8254</td>
<td>111.5696</td>
<td>-19.4886</td>
<td>236.8254</td>
<td>35.358</td>
<td>236.8254</td>
</tr>
</tbody>
</table>

Learning / Training Function

‘Trainlm’ from MATLAB toolbox is a network training function which updates weight and bias values according to Levenberg-Marquardt optimization [6]. It is the fastest back propagation algorithm in the MATLAB toolbox, and is immensely suggested as a first-choice supervised algorithm.

5. Result and Discussion

In this study, Neural Network Fitting Tool (using command: nftool) is used to create and train the network. The dataset is loaded into selected data window [6]. The network is trained using the input data and the performance plot, training state and regression plots are observed. In this training, Random (dividerand) rule divides the data where 70% data are assigned to training set, 15% to validation and 15% data to test set. As shown in Figure 4, this time the training continued for the maximum of 1000 iterations.

From the training state plot it is seen that training continued for iterations before the training stopped. The performance plot shown in Figure 6 does not indicate any major problems with the training. The validation and test curves are very similar.

6. Conclusion

Mostly mathematical models fail to simulate the complex nature of inverse kinematics problem. In contrast, ANN is based on the data input/output data pairs to determine the structure and parameters of the model. Also, ANN’s can always be updated in order to achieve better results by...
presenting new training examples as new data become available.

References


