Modeling and Identification of flexible link using Neural Networks

S. R. Rao and B. Bandyopadhyay
Systems and Control Engg.,
Indian Institute of Technology, Bombay,
Mumbai - 400 076, INDIA.
srao@rediffmail.com, bijnan@ee.iitb.ernet.in

B. Seth
Department of Mechanical Engg.,
Indian Institute of Technology, Bombay,
Mumbai - 400 076, INDIA.
seth@ee.iitb.ernet.in

Abstract - The challenge in the endpoint control of flexible manipulators is of obtaining an exact model of the nonlinear flexible system and then simulating it for further analysis. But being an infinite dimensional they pose problems in terms of predicting the behaviour and also exact mathematical representation. This problem of obtaining exact model can be solved if the behaviour of the flexible link system can be represented by a neural network, which would help further simulation, analysis and control thus eliminating the requirement of the exact mathematical representation of the system.

I. INTRODUCTION

Traditionally, robotic manipulators are designed and built in a manner that maximizes stiffness to minimize vibration and allow for good positional accuracy with relatively simple controllers. They are generally made of rigid and/or flexible links. Various advantages of the flexible links like high speed operation, lower energy consumption, high load to weight ratio and safer operation due to reduced inertia make them desirable to the rigid links. But the flexibility of such links make the position control of these links a difficult task because one has to consider not only the rigid modes but also the flexible modes of vibration.

Due to lightweight flexible links, the speed of manipulators may be increased, the material requirement and the power consumption reduced, and the overall cost may thus be minimized. However, very little of this idea has found its way into the industrial robot manipulators, since the associated mathematical model is highly nonlinear and complex, and the solution involves difficult and expensive computation. As a matter of fact, the actual flexible system properties are distributed in space, discrete models are a mere idealization... Unfortunately, because of the prominent non-linearity and complexity of mathematical models, exact solutions are impossible except for ideal cases. Approximate models which are generally discretisation in some way or the other are used.

A complete dynamic model of manipulator systems with flexibility is needed to predict the dynamic behavior for system design. A dynamic model will also provide insight for designing a controller to control lightweight manipulators at high speeds and heavy payloads. Comparison of Timoshenko and Bernoulli models shows that the more complicated Timoshenko model only improves results when the link is fairly rigid with fast positioning and high precision requirements [1].

Another type of analysis says, to form the basis of the manipulator analysis, the kinematics of the flexible-link manipulators can be described by means of an Equivalent Rigid Link System (ERLS) model [2]. As far as the link flexibility is concerned, the kinematic behavior of a flexible arm can be modeled using a finite number of functions which describe the link deformation [3].

The Lagrangian approach can be followed to derive the dynamic equations. For the rigid arms and elastic joints, several control algorithms using nonlinear static or dynamic state feedback have been proposed. These methods are essentially based on the input-output decoupling of the robot dynamic behavior. As a result, the closed-loop system turns out to be a set of linear and independent subsystems, so that standard techniques can be used for robot motion control. No specific restrictions on the deformability of the elastic links is imposed, e.g. the link is not required to be a planar beam. Apart from the above, the following work deals about the problem of modeling the flexible link. Also the problem of calculating the Inertia matrix of the manipulator is discussed in [4], [5]. It can be seen that most of the research deals with the linear models of the flexible manipulators which is far from the reality.

The nonlinear flexible link modeling is discussed in detail. The possibilities are analysed and dealt upon.
The problems are discussed and a neural network based method for obtaining such a model is discussed and analysed. The flexible link behaviour is obtained using an experiment wherein the system output is recorded for a particular input. This data is further used and a neural network model is constructed. This data is then validated and the results analysed. In section II the nonlinear model of the flexible link using the Timoshenko beam theory and Finite Element analysis are discussed and debated. In section III the experimental setup for generating the data for the flexible link is discussed. The modelling using neural networks is discussed in section IV. The factors which affect the learning time of the neural network are also discussed here.

II. NON-LINEAR MODEL OF SINGLE FLEXIBLE LINK

The dynamics of the flexible manipulator system can be described by a set of partial differential equations which represent a continuous system. This is effectively an infinite dimensional system. Two commonly used approximations to such a system include finite element models and modal expansion models.

The methodology is based upon Finite element and Lagrangian methods and the use of generalised Inertia matrix (GIM) concept. The single link manipulator system is modelled as being composed of links attached to each other with the first link attached to a fixed base [6]. Each link is assumed to be symmetrical about its longitudinal axis in the absence of deformation, and is considered an assemblage of distributed elements of equal lengths which are rigid in compression, but not in bending. Primary control inputs are applied at the joints and can be used to effect the movement of the end point of the link along feasible trajectories. Additional controls may be applied at intermediate points along the link for the purpose of damping the vibration modes and also to enhance the quality of control. Visous damping is introduced here apart from the centrifugal and Coriolis terms. Also, only the output of the actuator is considered and so the dynamics as well as the damping associated with the actuator system are not modeled.

A. Model using Finite Element approach

The overall approach involves treating the link as an assemblage of n elements of length l. For each element the kinetic energy $T_j$ and potential energy $V_j$ are computed in terms of suitably selected system of n generalised variables $q = [q_1 q_2 \ldots q_n]$ and their rate of change $\dot{q}$. These energies are then combined to obtain the total kinetic energy $T$ and potential energy $V$ for the entire system. That is,

$$T(q, \dot{q}) = \sum_{j=1}^{n} T_j, \quad V(q) = \sum_{j=1}^{n} V_j. \quad (1)$$

Knowledge of the kinetic and potential energies can be used to specify the Lagrangian

$$\mathcal{L}(q, \dot{q}) = T - V \quad (2)$$

The governing dynamic equations for the system can be derived through the Lagrange’s equations:

$$\frac{d}{dt} \left[ \frac{\partial \mathcal{L}}{\partial \dot{q}_k} \right] - \frac{\partial \mathcal{L}}{\partial q_k} = Q_k, \quad k = 1, 2, \ldots, n \quad (3)$$

where $Q_k$ are the generalised nonconservative forces.

Along with the boundary conditions these equations provide the desired dynamic form of the equations for the system as

$$M\ddot{q} + B\dot{q} - f = Q \quad (4)$$

where $M = M(q)$ is the nonlinear function of the GIM. $f = f(q, \dot{q})$ is vector of nonlinear functions of the generalised variables $q$ and $\dot{q}$. $B$ is the viscous friction term related with $\dot{q}$; and $Q$ represents torques applied at the joint. The above model poses certain problems related to the validation of the response. The research does not mention what parameters are considered for simulation and how the validation is carried out. Also the results listed are for the case of a two link flexible manipulator. Thus it becomes difficult to validate the model for the single flexible link, as here it is considered as there is no load at the tip, whereas in the two link model discussed in [6] the load of the second link and its dynamics may alter the behaviour of the tip. Hence, an attempt is made to construct the beam model based on Timoshenko beam theory.

B. Model using Timoshenko beam theory

In [7] the model using the Timoshenko beam theory is discussed as follows.

$$\begin{bmatrix} m_1 & m_2^T \\ m_2 & M_3 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} C_1 & 0 \\ 0 & C_2 \end{bmatrix} \begin{bmatrix} \dot{\theta} \\ \dot{q} \end{bmatrix} + \begin{bmatrix} 0 & 0 \\ 0 & -K_1 - \dot{\theta}^2 - C_2 \end{bmatrix} = \tau(t) \quad (5)$$

with net tip position output

$$y = [h \quad \Phi(h)] \begin{bmatrix} \theta \\ q \end{bmatrix} \quad (6)$$

$\tau(t)$ is the torque applied to the system at time $t$. The details of the other parameters and their significance in this model, can be found in [7]. This model
has been discussed with the payload \( M_f \) at the tip. Also, data required for generating the model is not sufficient. The major handicap here is, when one tries to calculate the structural damping constants and the Coulomb friction terms, one requires data relative to a particular link, which is missing.

III. Experiments

To study the exact behaviour of the nonlinear flexible link system it becomes necessary to conduct an experiment and obtain results based on certain inputs which indicate the system behaviour satisfactorily. The same approach is followed here and an experiment is conducted which helps in determining the input output behaviour.

A. Experimental setup

The flexible link with parameters as given in [8] is constructed and used for the experiment. Utmost care is taken to ensure that all the dimensions match so that it can be used for further analysis. The motor parameters are calculated and verified.

The experimental setup consists of a moving arrangement (generally a motor), a sensor arrangement (a piezoelectric accelerometer pickup) and a recording setup (Digital storage oscilloscope with FFT analyser). The piezo is placed at the point at which the output is to be recorded. The waveform at that point is recorded and its fast Fourier transform can be calculated which can be used for further analysis. This setup is sensitive to small changes in the input torque of the motor and the piezo sensor accurate enough to even small changes in the tip position. Using this setup the experiment is conducted and the output tip position is noted as \( y \). The torque \( \tau \) as shown in Fig.1 is applied and the corresponding output tip position is recorded with respect to time as shown in Fig. 2. These data are then further used for training the neural network.

From [7] it can be seen that the actual tip position determined experimentally is very much similar to the simulated one. So this response can be considered as the exact model of the single flexible link. Thus the model is validated and this response is used for further discussion and research.

IV. MODELLING USING ARTIFICIAL NEURAL NETWORKS

For nonlinear systems, in recent years a number of authors have addressed issues such as controllability, observability, feedback stabilisation, and observer design of nonlinear systems. Despite such attempts constructive procedures, similar to those available for linear systems, do not exist for nonlinear systems. Hence the choice of identification and controller models for nonlinear plants is a formidable problem and successful identification and control has to depend upon several strong assumptions regarding the input-output behaviour of the plant. In spite of the above comments, linear models described motivate the choice of structures or identifiers and controllers in the nonlinear case. It is in these structures that neural networks shall be incorporated.

Since very few results exist in nonlinear system theory, care has to be taken in the statement of problems, the choice of the identifier and controller structures, as well as generation of adaptive control laws. Neural networks can be used effectively for the modeling and identification of nonlinear dynamical systems [9].

The problem of characterisation is concerned with the mathematical representation of a system; a model of a system is expressed as an operator \( P \) from an input
space $\Phi$ into an output space $\Psi$ and the objective is to characterise the class $\Psi$ to which $P$ belongs. Given a class $\Psi$ and the fact that $P \in \Psi$ the problem of identification is to determine a class $\hat{\Psi} \subset \Psi$ and an element $P \in \hat{\Psi}$ so that $P$ approximates $P$ in some desired sense. In static systems $U$ and $Y$ are subsets of $\mathbb{R}^n$ and $\mathbb{R}^m$ respectively, while in dynamical systems they are generally assumed to be bounded Lebesgue integrable functions on the interval $[0, T]$ or $[0, \infty)$. In both the cases, the operator $P$ is defined implicitly by the specified input output pairs. The choice of the class of identification models $\hat{\Psi}$, as well as specific methods used to determine $P$, depends upon a variety of factors which are related to the accuracy desired, as well as analytical tractability.

The famous approximation theorem of Weierstrass states that an function in $C([a, b])$ can be approximated arbitrarily closely by a polynomial. A generalisation of Weierstrass’s theorem due to Stone, called the Stone-Weierstrass theorem can be used as the starting point for all the approximation procedures for dynamical systems [9]. Two classes of neural networks which have received considerable attention are:

1. Multilayer networks
2. Recurrent networks (feedback networks)

Both systems demand unified treatment because it is foreseen that dynamical elements and feedback will be increasingly used in the future rendering complex systems.

A. Training the Neural Network

Learning in the neural networks is a comparatively direct process where, every learning step can be captured as a distinct cause-effect relationship. To perform any task, neural network learning of an input-output mapping from a set of examples is needed. The learning task is to find out weights that provides the best possible approximation of a given function.

The neural networks can trained by either of the three ways mentioned below:

1. Supervised training
2. Unsupervised training

1. Supervised Training

In this type of training with each input pattern, the desired output is also provided to the neural network by the user (who uses his knowledge about the plant) and hence it is trained towards minimising the error between the desired output and neural network output. Typically, supervised learning rewards accurate classifications or associations and punishes those which yield inaccurate responses. Most supervised learning algorithms reduce to stochastic minimization of error.

2. Unsupervised Training

The neural network is given input training pattern and then it is allowed to generate its output (in which case the user lacks experience about the process and any beforehand knowledge of the plant is unavailable). The network is given some grading (performance index) as to how well it has been working and accordingly it adjusts the weights and biases.

In learning without supervision, the desired response is not known; thus explicit error information cannot be used to improve network behavior. Since no information is available as to correctness or otherwise, learning must somehow be accomplished based on the observations of the responses to inputs that we have marginal or no knowledge about.

B. Learning Rule

Learning rule is a method that shows, how a neural network is trained to modify its weights to give the desired output during its training.

The learning rules generally used are [10];

1. Delta learning rule
2. Hebbian learning rule
3. Widrow-Hoff learning rule
4. Correlation learning rule
5. Winner-take-all learning rule
6. Outstar learning rule

1. Levenberg Marquadt technique

Levenberg Marquadt technique is a more sophisticated method than gradient descent (backpropagation). This optimization technique is more powerful than gradient descent, but requires more memory. The Levenberg Marquadt update rule is

$$\Delta W = (J^T J + \mu I)^{-1} J^T e$$

where $J$ is the Jacobian matrix of derivatives of each error to each weight, $\mu$ is scalar, $e$ is the error vector. If the scalar $\mu$ is very large, this technique approximates the gradient descent, while it if it is small the above expression becomes the Gauss-Newton method. Because this method is faster, but tends to be less accurate when near an error minima, the scalar $\mu$ is adjusted like the adaptive learning rate. As long as the error gets smaller, $\mu$ is made bigger. But if the error increases $\mu$ is made smaller.
V. SYSTEM IDENTIFICATION AND VALIDATION

The output response is plotted and the data of input, output and time is used to train the multilayer neural network using MATLAB\textsuperscript{TM}. The trained model is then validated using validation data and the response is studied.

To train such a nonlinear system Multilayer Feedforward network with \textit{tansig} and \textit{purelin} transfer functions a combination which, with sufficient number of hidden nodes, capable of learning any type of system is used. Theoretically it is possible with such a type of system to approximate any type of complex polynomial function. The data generated in III. is used to train the multilayer neural network. The two layer neural network is used with \textit{tansig} and \textit{purelin} transfer functions. These type of transfer functions are used since this combination assures that the nonlinearities are covered. Also this two layer neural network has proved to be sufficient to learn any type of nonlinearity. This training of the system proves to be lengthy procedure.

\begin{center}
\includegraphics[width=\textwidth]{neural_network.png}
\end{center}

\textbf{Figure 3 – Neural network vs actual system response}

The weights and biases for the above system are given below

\begin{equation}
W_1 = \begin{bmatrix}
-1.6487 & 0.0820 & 0.0611 \\
1.1296 & -1.6286 & -0.8797 \\
2.8157 & 63.7663 & 4.6567 \\
-0.0068 & -0.1459 & 1.7965 \\
1.7830 & -0.1215 & 1.0309 \\
0.3015 & -0.2362 & -0.8271 \\
0.0395 & -3.7878 & -0.1166 \\
-0.1046 & 4.0778 & 0.167 \\
0.1889 & 3.9438 & 0.1851 \\
-0.7552 & -2.2165 & 1.5783
\end{bmatrix}
\end{equation}

\begin{equation}
B_1 = \begin{bmatrix}
3.3876 \\
-4.4004 \\
-9.9345 \\
-0.5974 \\
-16.0176 \\
0.4574 \\
-0.2119 \\
0.6907 \\
-1.1630 \\
7.4235
\end{bmatrix}
\end{equation}

\begin{equation}
W_2 = \begin{bmatrix}
w_1 & w_2
\end{bmatrix}
\end{equation}

where

\begin{equation}
w_1 = \begin{bmatrix}
-0.7873 & 0.5995 & 0.0243 & -7.4009 & 0.7915
\end{bmatrix}
\end{equation}

\begin{equation}
w_2 = \begin{bmatrix}
-6.8870 & -3.5254 & -2.3049 & -0.9944 & 1.6310
\end{bmatrix}
\end{equation}

\begin{equation}
B_2 = [0.5439]
\end{equation}

The \textit{tansig} transfer function is used in the hidden layer 1 and \textit{purelin} transfer function is used in the 2\textsuperscript{nd} layer. There has been research which considered some other type of neural networks which would use radial basis functions in lieu of the multi layer network. It was found while using the radial basis function networks not only take considerable time to learn, but also the number of neurons are not under the control of the designer. This depends upon the algorithm which is used to train. The number of hidden nodes thus affects the the later on as it controls the speed of simulation. Also when training is complete they contain very large number of neurons and this makes the system computationally expensive and time consuming. Radial basis networks can be simulated using SNNS. Training starts with an empty network and then the number of neurons goes on increasing. But this goes out of control for highly nonlinear systems as very large number of neurons also do not give satisfactory results. Also, radial basis networks can be effective only when interpolation is required. They have proved to be of little use in extrapolating the polynomial function/nonlinearity learnt. They can be effective used in applications such as pattern recognition and other types of cognition.

A. Learning Factors

Regarding neural networks we can generally follow three guidelines
1. Use one hidden layer.
2. Use very few hidden neurons.
3. Train until you can.

With larger number of layers the problem is that the training often slows dramatically. This is due to two effects:
1. The additional layer through which errors must be backpropagated makes the gradient more unstable.
2. The number of false minima increases dramatically.

The back-propagation learning algorithm in which synaptic weights are systematically modified so that the response of the network increasingly approximates the desired response can be interpreted as an optimisation problem. The learning algorithm modifies the weight matrices so that the error value decreases. It might appear that the error back-propagation has made a breakthrough in supervised learning of the layered networks. In practice, however, implementation of the algorithm may encounter different difficulties which are:

- Initial Weights
- Cumulative weight adjustment versus Incremental Updating
- Steepness of the Activation Function
- Learning Constant
- Momentum and other methods
- Network Architecture
- Number of Hidden neurons

VI. CONCLUSION

An experiment is conducted on the single flexible link to study its nonlinear behaviour. The data obtained from the experiment is further learnt using a multilayer neural network to build a nonlinear model. This model is further validated for the test data. Thus the neural network network representation of the flexible link is successfully implemented and tested.

REFERENCES


