A Virtual Environment for Studying Flexible Robot Manipulators

doi:10.3991/ijoe.v5i4.918

A.K.M. Azad[1], M.O. Tokhi[2], and M.H. Shaheed[2]
[1] Northern Illinois University, USA.
[2] University of Sheffield, UK.
[3] University of London, UK.

***Abstract*—The paper presents a virtual environment for simulation and modeling a single-link flexible robotic manipulator. Simulation algorithm is based on the finite difference method where the systems governing dynamic equation is discretized and implemented. Intelligent modeling techniques are developed using neural network and genetic algorithm based techniques. All these have been developed and realized within a virtual environment using the Matlab, Simulink, and associated toolboxes. An interactive user friendly graphical user interface has also been presented. This environment has proven to be a valuable educational tool for understanding the behavior of flexible manipulator systems and can also be used as a computer aided teaching facility and a testbed for controller designs.**

***Index Terms*—Intelligent systems, flexible robot manipulator, system modeling, virutal environment, and simulation.**

I. INTRODUCTION

Considering the advantages of flexible robot manipulators, some of the current applications of such manipulators include spacecraft, remote manipulation, and radioactive material handling in nuclear power plants [1]. However, due to their flexible nature, induced vibrations appear in the system during and after a positioning motion [2-3]. This restricts their widespread use in industry. A considerable amount of research work has already been carried out on the vibration control of flexible robot manipulators. However, a generic solution to the problem is yet to be obtained [4-5].

To formulate and implement an effective control strategy for efficient vibration suppression of the system, it is essential to develop a mathematical model for the system that accounts for the interactions with actuators and payload [6]. Such a model can be constructed by solving the partial differential equations (PDEs) that describe the system. However, the computational complexity and subsequent software coding involved in the process is a major disadvantage of this technique [7]. An alternative solution is to utilize intelligent techniques, such as neural network (NN) and genetic algorithm (GA) for modeling of flexible robot manipulator systems [8]. The approaches have proven to be effective in modeling and can be used for test and verification of controller designs.

As far as this authors' knowledge there is no interactive virtual environment that will allow one to study the behavior of flexible robot manipulators without going into the software coding details. To address this problem, the authors have developed a virtual environment that will allow the users to study the behavior of flexible robot manipulators through classical simulation and intelligent modeling techniques. The environment is developed by using Matlab, Simulink and few associated toolboxes.

The rest of the paper is structured as follows: Section 2 briefly describes the flexible robot manipulator system considered for this development. Section 3 presents the theoretical analysis of finite difference (FD) simulation, NN modeling, GA modeling, and subsequent model validation strategies. Section 4 describes the developed virtual environment with some examples. The paper is concluded in Section 5.

II. THE FLEXIBLE ROBOT MANIPULATOR SYSTEM

The flexible robotic manipulator system under consideration is modeled as a pinned-free flexible beam, with a mass at the hub, which can bend freely in the horizontal plane but is stiff in vertical bending and torsion. The model development utilizes the Lagrange equation and modal expansion method [9-10]. To avoid the difficulties arising due to time varying length, the length of the manipulator is assumed to be constant.

A schematic representation of the manipulator is shown in Figure 1, where X_oOY_o and XOY represent the stationary and moving co-ordinate frames, respectively. The axis OX coincides with the neutral line of the link in its undeformed configuration and is tangent to it at the clamped end in a deformed configuration. The τ represents the applied torque at the hub. E, I, ρ, S, I_h and m_p represent the Young modulus, area moment of inertia, mass density per unit volume, cross sectional area, hub inertia and payload of the manipulator, respectively. $\theta(t)$ denotes an angular displacement (hub-angle) of the manipulator and $w(x,t)$ denotes an elastic deflection (deformation) of a point along the manipulator at a distance x from the hub of the manipulator. In this work, the motion of the manipulator is confined to the X_oOY_o plane. The width of the arm is assumed to be much greater than its thickness, thus, allowing the manipulator to vibrate (be flexible) dominantly in the horizontal direction. The shear deformation and rotary inertia effects are also ignored.

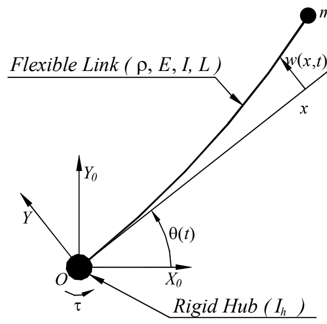


Figure 1. Schematic representation of the flexible robot manipulator system.

For an angular displacement θ and an elastic deflection u, the total (net) displacement $y(x,t)$ of a point along the manipulator at a distance x from the hub can be described as a function of both the rigid body motion $\theta(t)$ and elastic deflection $w(x,t)$ measured from the line OX_0;

$$y(x,t) = x\theta(t) + w(x,t) \tag{1}$$

To obtain equations of motion of the manipulator, the associated energies have to be obtained. These include the kinetic, potential, and dissipated energies. To obtain equations of motion of the manipulator, the associated energies have to be obtained. These include the kinetic, potential, and dissipated energies. Thus, using the Hamiltonian's extended method, the dynamic equation of the flexible robot manipulator with the associated boundary and initial conditions can be expressed as [2]:

$$EI\frac{\partial^4 \mathrm{y(x,t)}}{\partial x^4} + \rho\frac{\partial^2 \mathrm{y(x,t)}}{\partial t^2} - D_s\frac{\partial^3 \mathrm{y(x,t)}}{\partial x^2 \partial t} = \tau(\mathrm{t}) \tag{2}$$

$$\begin{gathered} y(0,t) = 0 \\ I_h\frac{\partial^3 y(0,t)}{\partial t^2 \partial x} - EI\frac{\partial^2 y(0,t)}{\partial x^2} = \tau(t) \\ M_p\frac{\partial^2 y(l,t)}{\partial x^2} - EI\frac{\partial^3 y(l,t)}{\partial x^3} = 0 \\ EI\frac{\partial^2 y(l,t)}{\partial x^2} = 0 \end{gathered} \tag{3}$$

$$y(x,0) = 0, \qquad \frac{\partial y(x,0)}{\partial x} = 0 \tag{4}$$

where, E is the Young modulus, I is the second moment of inertia of the manipulator, and $\tau(x,t)$ is the applied torque. For the system under consideration, the torque $\tau(x,t)$ is applied at the hub of the manipulator, therefore, it can be represented as $\tau(0,t)$ or simply $\tau(t)$.

III. SIMULATION AND INTELLIGENT MODELING

The system modeling techniques using NN and GA require input output(s) data of a system for their model development. To address this issue, a simulation algorithm has been developed to provide required input output(s) data for a user specified flexible robot manipulator system. A FD algorithm has been used to develop this simulation. The accuracy of the simulation outcome can be adjusted by changing the number of grid points along the length of the manipulator.

A. FD simulation

The PDE in equation (2) describing the flexible robot manipulator system is of a hyperbolic type and can be classified as a boundary value problem. This can be solved using an FD method [11]. This involves dividing the arm into a finite number of equal-length sections and developing a linear difference equation describing the deflection of end of each section (grid-point). Thus, using the FD method, a solution of the PDE in equation (2) can be obtained as [2];

$$\begin{aligned} y_{i,j+1} = {} & \frac{\Delta t^2}{\rho}\tau(i,j) - c\left[y_{i+2,j} + y_{i-2,j}\right] + b\left[y_{i+1,j} + y_{i-1,j}\right] + ay_{i,j} - y_{i,j-1} \\ & + d\left[y_{i+1,j} - 2y_{i,j} + y_{i-1,j} - y_{i+1,j-1} + 2y_{i,j-1} - y_{i-1,j-1}\right] \end{aligned} \tag{5}$$

where

$a = 2 - 6c$; $b = 4c$; $c = EI\Delta t^2/\left(\rho\Delta x^4\right)$ and $d = D_s\Delta t/\left(\rho\Delta x^2\right)$.

Equation (5) gives the displacement of section i of the manipulator at time step $j+1$. It follows from this equation that, to obtain the displacements $y_{n-1,j+1}$ and $y_{n,j+1}$, the displacements of the fictitious points $y_{n+2,j}$, $y_{n+1,j}$ and $y_{n+1,j-1}$ are required. These fictitious displacements can be obtained using the boundary conditions related to the dynamic equation of the flexible robot manipulator. The discrete forms of the corresponding boundary conditions, obtained in a similar manner as above are

$$y_{0,j} = 0 \tag{6}$$

$$y_{-1,j} = y_{1,j} + \frac{\Delta x I_h}{EI\Delta t^2}\left[y_{1,j+1} - 2y_{1,j} + y_{1,j-1}\right] + \frac{\Delta t^2}{EI}\tau(j) \tag{7}$$

$$y_{n+2,j} = 2y_{n+1,j} - 2y_{n-1,j} + y_{n-2,j} + \frac{2\Delta x^3 M_p}{\Delta t^2 EI}\left[y_{n,j+1} - 2y_{n,j} + y_{n,j-1}\right] \tag{8}$$

$$y_{n+1,j} = 2y_{n,j} - y_{n-1,j} \tag{9}$$

Manipulating equation (5) using equations (6) to (9) yields a matrix formulation of the above as

$$\mathbf{Y}_{i,j+1} = \mathbf{AY}_{i,j} + \mathbf{BY}_{\mathrm{i,j-1}} + \mathbf{CF} \tag{10}$$

where $\mathbf{Y}_{i,k}$ is the displacement of grid points $i = 1,2,\cdots,n$ of the manipulator at time step k ($k = j+1, j, j-1$). **A** and **B** are constant n x n matrices whose entries depend on the flexible robot manipulator specification and the number of sections the manipulator is divided into, **C** is a constant matrix related to the time step Δt and mass per unit length of the manipulator and **F** is an n x 1 matrix related to the given input torque [2]. Equation (10) thus represents the dynamic equation of the manipulator in the presence of hub-inertia and payload, which can easily be implemented within the Matlab and Simulink environments.

B. Intelligent modeling

In many cases, when it is difficult to obtain a model structure for a system with traditional system identification techniques, intelligent techniques are desired that can describe the system in the best possible way [11]. NN and GA are two intelligent techniques used commonly for system identification and modeling. The major advantage of utilizing GA for system identification is that GA simultaneously evaluates many points in the parameter space and converges towards the global solution [12]. The superiority of a GA over recursive least squares (RLS) algorithm in modeling a fixed-free flexible beam is addressed by [13]. In contrast, NN approaches for system identification offer many advantages over traditional ones, especially in terms of flexibility and hardware realization [14]. This technique is quite efficient in modeling non-linear systems or if the system possesses nonlinearities to any degree.

Once a model of the system is obtained, it is required to validate whether the model is good enough to represent the system. A number of such validation tests are available in the literature [14]. These are one-step ahead (OSA) prediction; model predicted output (MPO), and correlation tests. Such techniques are incorporated within SCEFMAS for validating developed models. Moreover, with NN modeling, the input-output data set is divided into two halves. The first half is used to train the NN and the output computed. The NN usually tracks the system output well and converges to a suitable error minimum. New inputs are presented to the trained neural network and the predicted output is observed. If the fitted model is correct, i.e., correct assignment of lagged inputs and outputs, then the network will predict well for the prediction set. In this case, the model will have captured the underlying dynamics of the system. If both the OSA and the MPO of a fitted model are good over the estimation and prediction data sets, then most likely the model is unbiased.

NN modeling: Various modeling techniques can be used with neural networks to identify a non-linear dynamical system. These include state-output model, recurrent state model, and non-linear autoregressive moving average process with exogeneous (NARMAX) input model. However, from the literature, it has been established that if the plant's input and output data are available, the NARMAX model is a suitable choice, for modeling systems with nonlinearities. Mathematically, the model is given as [15-16]:

$$\begin{aligned}\hat{y}(t) = f[(&y_p(t-1), y_p(t-2), \cdots, y_p(t-n_y),\\ &u(t-1), u(t-2), \cdots, u(t-n_u),\\ &e(t-1), \mathrm{e}(t-2), \cdots, e(t-n_e)] + e(t)\end{aligned} \tag{11}$$

Where, $\hat{y}(t)$ is the output vector determined by the past values of the system input vector, output vector and noise with maximum lags n_y, n_u and n_e, respectively, $f(\cdot)$ is the system mapping constructed through multi-layer perceptron or radial basis function neural networks with an appropriate learning algorithm. The model is also known as NARMAX equation error model. However, if the model is good enough to identify the system without incorporating the noise term or considering the noise as additive at the output, the model can be represented in a NARX form [15-17] as:

$$\begin{aligned}\hat{y}(t) = f[(&y_p(t-1), y_p(t-2), \cdots, y_p(t-n_y),\\ &u(t-1), u(t-2), \cdots, u(t-n_u)] + e(t)\end{aligned} \tag{12}$$

The system is shown in Figure 2.

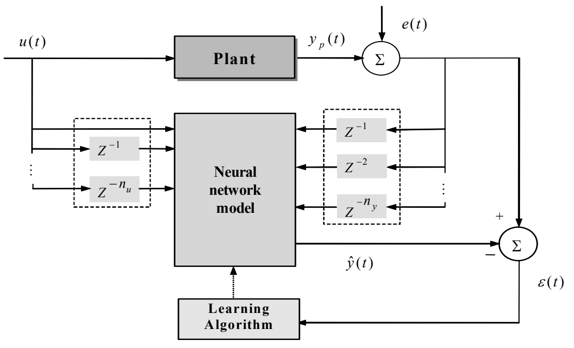


Figure 2. NARX model identification with MLP/RBF neural networks.

GA modeling: With the GA modelling, an initial population of potential solutions is created in the first step. In the second step, the performance of each member of the population is assessed through an objective function imposed by the problem. This establishes the basis for selection of pairs of individuals that will be mated together during reproduction. For reproduction, each individual is assigned a fitness value derived from its raw performance measure, given by the objective function [18].

In the manipulation phase, genetic operators such as crossover and mutation are used to produce a new population of individuals (offspring) by manipulating the genetic information, usually called genes, possessed by the members (parents) of the current population. In this way, the average performance of individuals in a population is expected to increase, as good individuals are preserved and breed with one another and the less fit individuals die out. The GA is terminated when some criteria are satisfied, e.g., a certain number of generations completed or when a particular point in the search space is reached.

For parametric identification of the manipulator with GA, randomly selected parameters are optimised for different arbitrarily chosen order to fit to the system by applying the working mechanism of GA as described above. The fitness function utilised is the sum-squared error between the actual output, $y(n)$, of the system and the predicted output, $\hat{y}(n)$, produced from the input to the system and the optimised parameters:

$$f(e)=\sum_{i=1}^{n}\left(\left|y_p(n)-\hat{y}(n)\right|\right)^2 \tag{13}$$

Where, n is the number of input/output samples. With the fitness function given above, the global search technique of the GA is utilized to obtain the best set of parameters among all the attempted orders for the system. The output of the system is thus simulated using the best sets of parameters and the system input.

Model validation: It is important to devise an effective model validation process once a model has been developed. There are various methods of validating a model, such as: one step ahead (OSA) prediction, model predicted output (MPO), and correlation tests. Both the MPO and correlation tests are implemented for this virtual environment. The MPO method can be expressed as:

$$\hat{y}(t)=f(u(t),u(t-1),....,u(t-n_u),\hat{y}(t-1),....,\hat{y}(t-n_y)) \tag{14}$$

and the deterministic error or deterministic residual is

$$\varepsilon(t)=y_p(t)-\hat{y}(t) \tag{15}$$

If only lagged inputs are used to assign network input nodes, then

$$\hat{y}(t)=\hat{y}_d(t) \tag{16}$$

The implication that the fitted model behaves well for the MPO does not necessarily imply that the model is unbiased. The prediction over a different set of data often reveals that the model could be significantly biased. One way to overcome this problem is by splitting the data set into two sets, an estimation set and a test set (prediction set). A section of the data is used for training the network and the remaining portion of the data for validating and testing the network.

A more convincing method of model validation is to use correlation tests. If a model of a system is adequate, then the residuals or prediction errors $\varepsilon(t)$ should be unpredictable from all linear and non-linear combinations of past inputs and outputs. The derivation of simple tests, which can detect these conditions, is complex, but it can be shown that the following conditions should hold [19].

$$\begin{aligned}
&\phi_{\varepsilon\varepsilon}(\tau)=E[\varepsilon(t-\tau)\varepsilon(t)]=\delta(\tau)\\
&\phi_{u\varepsilon}(\tau)=E[u(t-\tau)\varepsilon(t)]=0 \qquad \forall\tau\\
&\phi_{u^2\varepsilon}(\tau)=E[(u^2(t-\tau)-\bar{u}^2(t))\varepsilon(t)]=0 \qquad \forall\tau \ldots (17)\\
&\phi_{u^2\varepsilon^2}(\tau)=E[(u^2(t-\tau)-\bar{u}^2(t))\varepsilon^2(t)]=0 \qquad \forall\tau\\
&\phi_{\varepsilon(\varepsilon u)}(\tau)=E[\varepsilon(t)\varepsilon(t-1-\tau)u(t-1-\tau)]=0 \qquad \tau\ge 0
\end{aligned}$$

where $\phi_{u\varepsilon}(\tau)$ indicates the cross-correlation function between $u(t)$ and $\varepsilon(t)$, $\varepsilon u(t)=\varepsilon(t+1)u(t+1)$, $\delta(\tau)$ = an impulse function.

Ideally the model validity tests should detect all the deficiencies in network performance, including bias due to internal noise. The cause of the bias will however be different for different assignments of network input nodes. Consequently, the full five tests defined by equation (11) should be satisfied if u(.)'s and y(.)'s are used as network input nodes. In practice normalized correlations are computed. The sampled correlation function between two sequences $\psi_1(t)$ and $\psi_2(t)$ is given by,

$$\hat{\phi}_{\psi_1\psi_2}(\tau)=\frac{\sum_{t=1}^{N-\tau}\psi_1(t)\psi_2(t+\tau)}{\sqrt{\sum_{t=1}^{N}\psi_1^{\,2}(t)\sum_{t=1}^{N}\psi_2^{\,2}(t)}} \tag{18}$$

Normalization ensures that all the correlation functions lie in the range $-1\le\hat{\phi}_{\psi_1\psi_2}(\tau)\le 1$, irrespective of the signal strengths. The correlations will never be exactly zero for all lags, and the 95% confidence bands defined as $1.96/\sqrt{N}$ are used to indicate if the estimated correlations are significant or not, where N is the data length. Therefore, if the correlation functions are within the confidence intervals, the model is regarded as adequate.

IV. THE VIRTUAL ENVIRONMENT

The development is composed of two main components, one is the Matlab (and associated toolboxes) driven computing part and the other is the Guide driven front panel known as GUI [20-22].

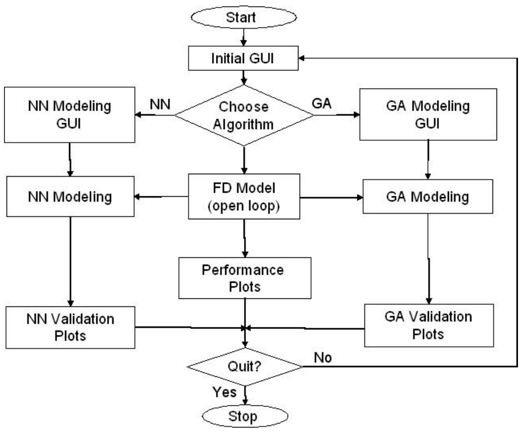


Figure 3. A flowchart of the environment.

The user provides the system specification through a GUI that is subsequently passed to the Matlab for computation. The computation outcomes are then passed to other GUIs for presentation of the results. A flowchart representing the environments' structure is shown in Figure 3.

There are three main parts of the computing component: a) FD simulation; b) NN modeling and validation; and c) GA modeling and validation. As the name implies the FD simulation part provides the computation for FD simulation of a flexible robot manipulator system. While the NN modeling and validation and GA modeling and validation parts provide computation for NN and GA modeling and subsequent model validation, respectively. The training data used for NN and GA modeling processes are obtained from an open-loop FD simulation, using random or composite PRBS torque inputs.

The GUI component has six parts: a) Initial GUI; b) Results GUI; c) NN modeling GUI; d) GA modeling

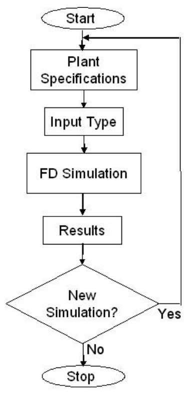


Figure 4. Flowchart for the FD simulation implementation.

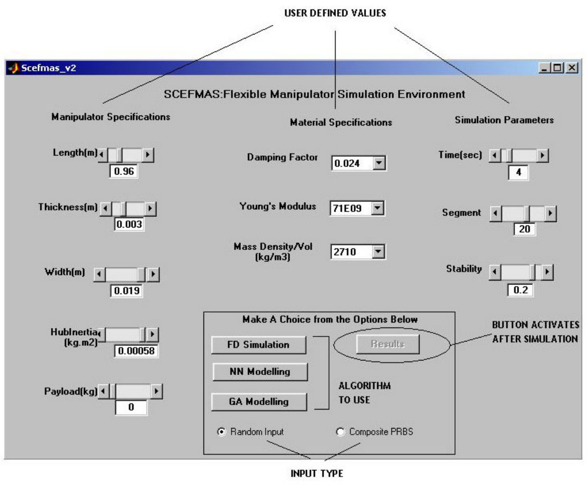
GUI; e) NN model validation GUI; and f) GA model validation GUI. Initial GUI is the home of the SCEFMAS and is used to specify a desired flexible robot manipulator, FD simulation parameters, nature of excitation input, and direct the users towards a desired intelligent modeling technique. Results GUI is used to display the FD simulation input and output both in time and frequency domains along with a three dimensional display of the system movement. The NN modeling GUI allows the users to provide the NN modeling structure and also guides them through the NN modeling process. The GA modeling GUI allows the users to pass the GA modeling parameters and also guides them through the GA modeling process. NN and GA model validation GUIs are used to present the validation graphs after NN and GA modeling, respectively.

A. The FD simulation and verification

The FD simulation is one of the main parts of the environment. This provides an open-loop simulation of a specified flexible robot manipulator system.

The output data generated through this simulation process along with the excitation inputs are stored and subsequently used for NN and GA modeling. The input types, which have been provided within the environment, are the random and composite PRBS. The implementation flowchart for the FD simulation is shown in Figure 4.

Within the Initial GUI a user provides the desired system specification, simulation parameters, and input types (Figure 5). The system specification consists of: <u>Manipulator Specifications</u>, <u>Material Properties</u>, and <u>Simulation Parameters</u>. Manipulator specifications involve the length, thickness, and width of the manipulator along with the hub inertia and payload. While the material properties constitute a damping factor, Young's modulus, and mass density per volume. Finally, the total simulation time, number of segments, and stability factor for the FD algorithm constitute the simulation parameters. At the bottom of the option selection box, there are two radio buttons, which can be used to select excitation input of the manipulator. The inputs are *Random* input and *Composite PRBS* input. Only these kinds of inputs provide sufficient excitation for all the modes associated with a flexible robot manipulator system within the frequency range of excitation. After providing all the information within the initial GUI, one can click on the FD simulation button within the GUI. This will bring up a pre-developed Simulink model of the flexible robot manipulator system connected in an open-loop manner.



SCEFMAS:Flexible Manipulator Simulation Environment

Manipulator Specifications

Length(m) 0.96
Thickness(m) 0.003
Width(m) 0.019
HubInertia (kg.m2) 0.00058
Payload(kg) 0

Material Specifications

Damping Factor 0.024
Young's Modulus 71E09
Mass Density/Vol (kg/m3) 2710

Simulation Parameters

Time(sec) 4
Segment 20
Stability 0.2

Make A Choice from the Options Below

[✓ Random Input] Composite PRBS

Figure 5. Initial GUI for the SCEFMAS environment.

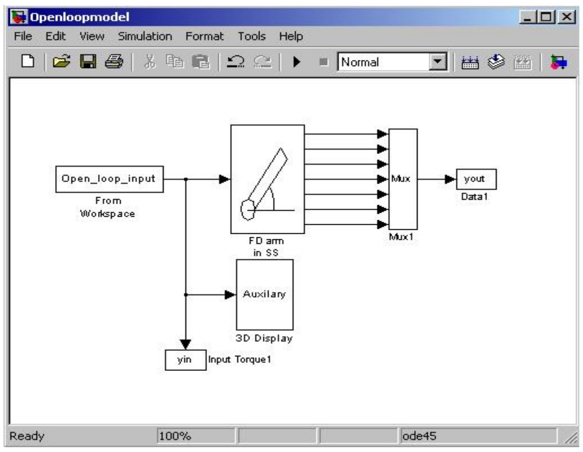


Figure 6. A Simulink model of a flexible robotic manipulator connected in an open-loop manner.

The model consists of a `FD arm in SS` block, which implements the FD algorithm for the flexible robot manipulator in state-space form. The *composite PRBS* torque input is provided from the Matlab workspace through the `Open_loop_input` block. Along with the `FD arm in SS` block, this torque input is also passed to the `Auxiliary` and `yin` blocks. The `Auxiliary` block produces data for 3D displacement of the manipulator, while the `yin` block passes the input torque values to the Matlab workspace for further analysis. The output of the `FD arm in SS` block contains the displacement, velocity, and acceleration data for the hub point and end point of the flexible robot manipulator. These data are passed to the Matlab workspace through the `Mux` and `yout` block. A simulation run of the Simulink model will produce outputs and will be passed to the Matlab workspace along with excitation input data. After the completion of the simulation run, a new button will appear within the Initial GUI, next to the `FD Simulation` button. This button is called the `Results` button (Figure 5). A click on this button will open the Results GUI that can be used to display the input and all the outputs produced through this simulation process (Figure 7).

The left hand side of the Results GUI includes option buttons, the top right part contains time domain and frequency domain result windows fo r the selected input or

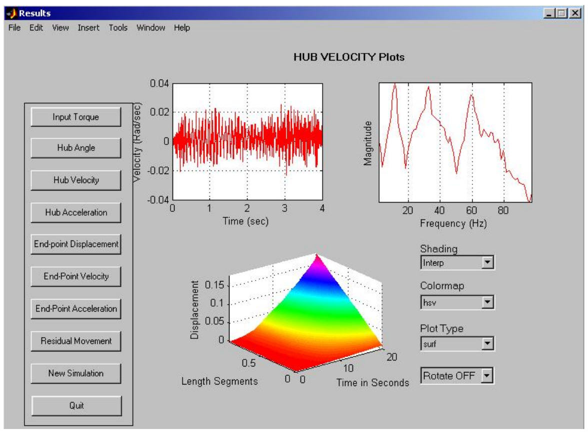


Figure 7. Displaying the results obtained from FD Simulation

output parameter, bottom middle window is for displaying a 3D plot of the complete motion of the manipulator for a given simulation run. The drop down menus at the bottom right side can be used to choose properties of the 3D plot. After viewing the result, the user can choose to return to the Initial GUI by clicking on the New Simulation button or may exit the environment by clicking on the Quit button within the Results GUI (Figure 7).

B. B. NN modeling and validation

The NN Modeling GUI is used to carry out the NN modeling of a flexible robot manipulator system. The implementation flowchart for the NN modeling and subsequent model validation is shown in Figure 8.

This NN Modeling GUI can be invoked by clicking on the NN Modeling button within the Initial GUI (Figure 9). The user can choose three, four, or five layers of neurons. The *types of neurons* along with the *number of neurons* in each layer can also be selected. For this specific case, a three layer structure was chosen with 5 neurons in each of the inside layers.

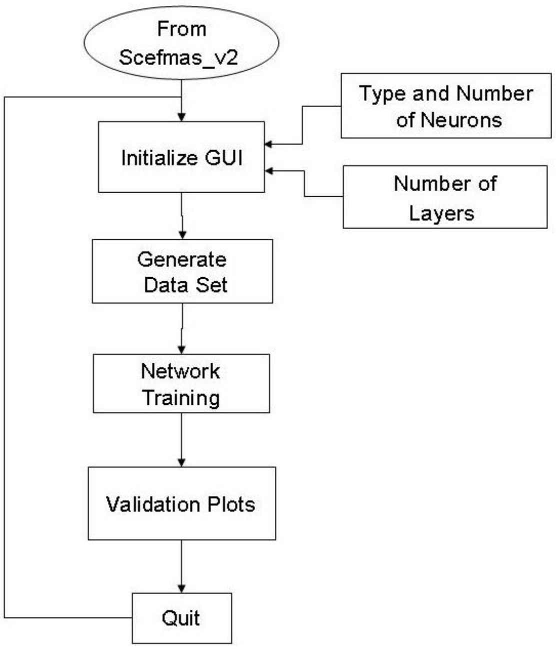


Figure 8. Flowchart for NN modeling strategy.

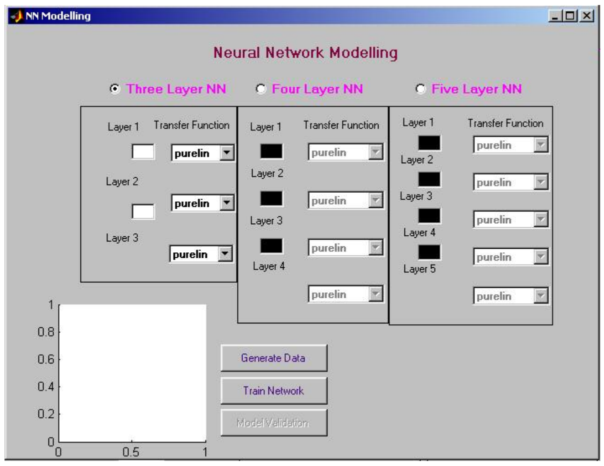


Figure 9. The GUI used for neural network modeling.

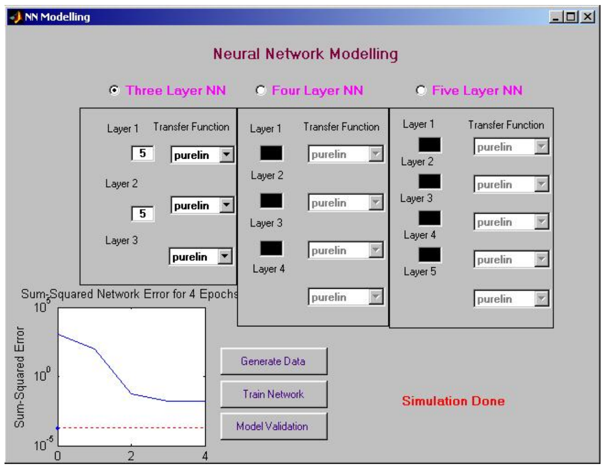


Figure 10. NN modeling GUI at the end of a simulation run.

After specifying the desired NN structure, the user can use the available input-output data set from a previous run or generate a new set of training data by clicking on the Generate Data Set button (Figure 9). This will open a Simulink model with open-loop input as shown in Figure 6. A subsequent run of the open-loop FD simulation model will produce the input-output data necessary for the NN training.

After having training data, the user needs to click on the Train Network button within the NN Modeling GUI. This will start the NN training process (Figure 10). The progress through the training can be monitored through the graph window shown on the left hand side of the GUI. The graph will plot the sum-squared error. At the completion of the training process, a message Simulation Done will appear within the NN Modeling GUI. This will be followed by the appearance of another button called the Model Validation button.

The Model Validation button will allow the users to examine the quality of the developed NN model by observing all the three model outputs (input to hub angle, input to hub velocity, and input to end-point acceleration). The model validation process involves the MPO and correlation tests that are presented through the NN Model Validation GUI. The GUI with MPO validation plots for a hub angle model in the time and frequency domains are shown in Figure 11.

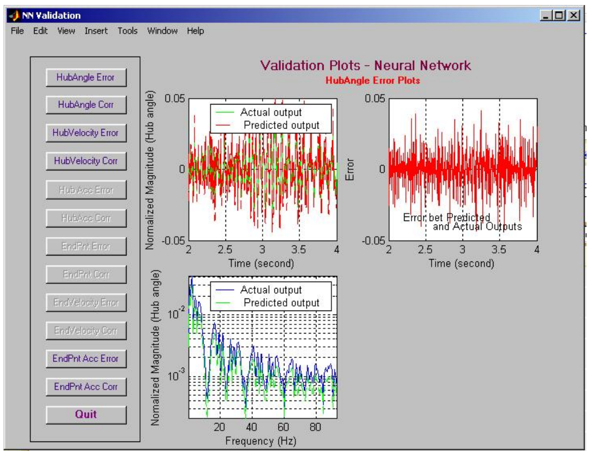


Figure 11. Time and frequency domain validation plots for a developed NN model.

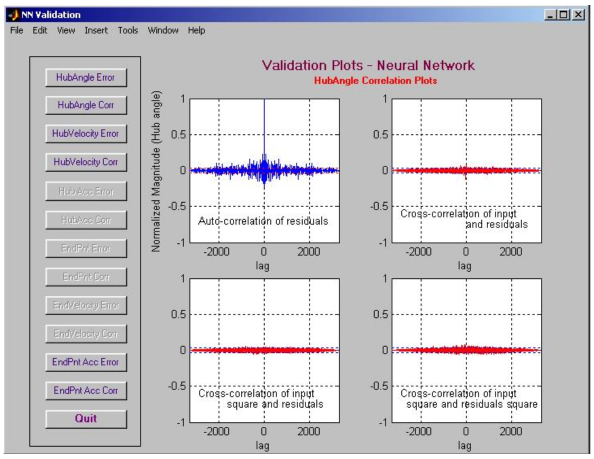

NN Validation
File Edit View Insert Tools Window Help
Validation Plots - Neural Network
HubAngle Correlation Plots
HubAngle Error
HubAngle Corr
HubVelocity Error
HubVelocity Corr
Hub Acc Error
Hub Acc Corr
EndPnt Error
EndPnt Corr
EndVelocity Error
EndVelocity Corr
EndPnt Acc Error
EndPnt Acc Corr
Quit
Normalized Magnitude (Hub angle)
Auto-correlation of residuals
Cross-correlation of input and residuals
Cross-correlation of input square and residuals
Cross-correlation of input square and residuals square
lag

Figure 12. Validation plots for a developed hub angle model through correlation tests.

The left hand side of the GUI provides all the model validation option buttons. The plots within the GUI show a comparison between the actual and model outputs for input to hub angle model. The top left corner graphical window shows the normalized actual and predicted outputs, while the top right corner graphical window shows the error between the actual (from FD simulation) and NN model output. The bottom graph window compares the actual and predicted outputs in the frequency domain. All the graphs show close matches between the real and model predicted outputs. The performance of models from the input to the remaining two outputs can be displayed by clicking on the appropriate button. In addition to the MPO validation plots, one can obtain correlation test plots for each of the three model outputs. The hub-angle validation plots with correlation tests are shown in Figure 12. The auto-correlation and cross-correlation graphs show close proximity between the real output and predicted output obtained from the NN model. The user can click on the Quit button to return to the Initial GUI and start simulation all over again with a different set of system parameters.

C. *GA modeling and validation*

This section describes the GA modeling and model validation process. The user will have the choice of specifying GA model parameters, such as number of individuals, maximum number of generations, generation gap, binary precision, and the order of the GA model.

Similar to the NN modeling, the input and outputs from a FD simulation model will be used as a reference for the model development process. A flowchart describing the GA modeling steps is shown in Figure 13. The quality of the developed model can be verified both in time and frequency domains along with suitable correlation tests.

The GA Modeling GUI is used to carry out the GA modeling of a flexible robot manipulator system (Figure 14). This GUI can be invoked by clicking on the GA Modeling button within the Initial GUI (Figure 5). The left hand side of the GUI is provided with sliders, where a user can set the GA modeling parameters. These are *Number of individuals*, *Maximum number of generations*, *Generation gap*, *Binary precision,* and *Order of GA model*. The top right corner of the GUI provides a drop

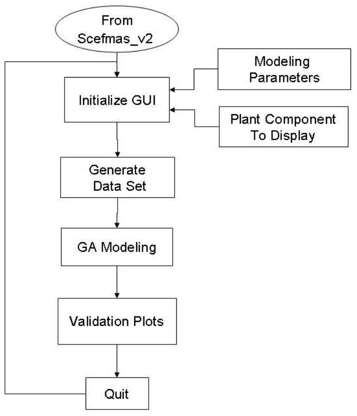


Figure 13. Flowchart for GA modeling process.

down menu to choose a model type, such as input to *Hub Angle*, input to *Hub Velocity,* and input to *End-point Acceleration*. After entering all the model parameters and model types, the user can generate training data by clicking on the Generate Data button (one of the right bottom buttons) or can use the previously generated simulation data available within the system. The user can proceed with the GA modeling by clicking on the GA Modeling button within the GUI. The progression through the GA modeling along with the fitness performance will be displayed in the figure window at the middle of the GUI (Figure 14).

After the completion of the modeling process, the Validation Plots button will be activated (bottom right corner of the GUI). The user can click on this button to observe the performance of the developed GA model through GA Validation GUI (Figure 15). The left hand side of the GUI is provided with buttons for choosing model types. The model validation outputs are displayed through four figure windows. The top two figure windows are for comparing the magnitudes of actual and predicted outputs. The bottom right window shows the comparison between actual and predicted outputs in the frequency domain. The bottom left window shows the sum-squared error for each generation. All the validation plots show a close match between the real data and the output produced by the develop GA model. At the end of the GA model validation process, the user can return to the Initial GUI for further modeling exercises.

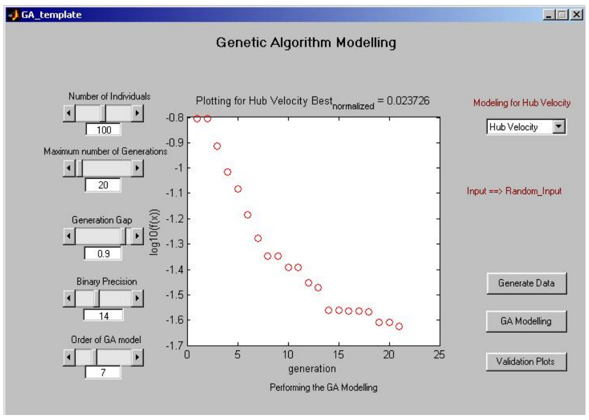


Figure 14. The GUI used for GA modelling process.

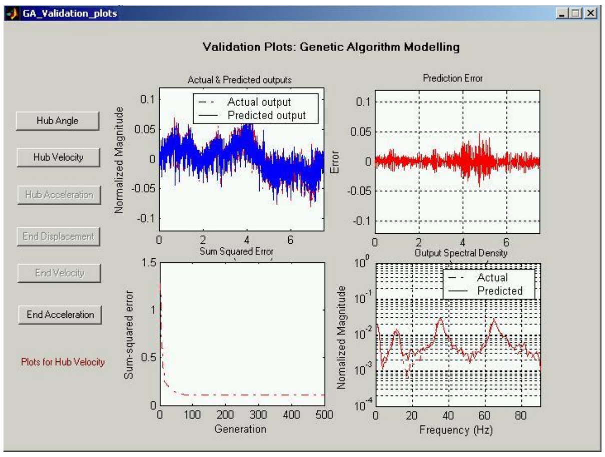


Figure 15. GA modeling validation GUI.

V. CONCLUSION AND DISCUSSION

The FD based simulation of a single-link flexible robotic manipulator is the backbone of the environment. The user can specify a desired flexible robot manipulator system and observe the behavior of the system (both in time and frequency domains) before moving into the intelligent modeling process. As a part of intelligent modeling techniques, NN and GA models have been developed and realized using the Matlab, Simulink, Guide, and other associated toolboxes. Results of various modeling techniques have been validated through various tests, including input/output mapping, training and test validation, and correlation tests. The interactive GUIs allow the user to choose a model structure and monitor the developed model performance without going into the programming details. Moreover, a data analysis provision has been made within the package to enable users to analyze data obtained from a test run. This makes the environment more user-friendly and saves the time and effort to transfer the data to another environment for analysis.

The environment is in use with the Automatic Control and Systems Engineering Department of The University of Sheffield (UK). This package is used as a supporting tool to deliver a module of a M.Sc. program within the department. The virtual environment enables the students to understand the behavior of a flexible robot manipulator system and also the effect of parameter variations. The learning process could be much more difficult without this package. In addition to this, students can test the effectiveness of their controller designs without spending much time on system modeling. The addition of these new features will enhance the learning outcomes from this environment.

The NN and GA modeling features can easily be extended to the development of intelligent controllers to investigate various aspects of active vibration control in flexible robot manipulator systems. With the advent of Internet technology and the Matlab web server, the package can be further developed to be used as a distance learning facility.

REFERENCES

[1] Book, W. J. and Majette, M. (1983). Controller design for flexible distributed parameter mechanical arms via combined state-space and frequency domain techniques, *Transaction of ASME Journal of Dynamic Systems, Measurement and Control*, **105**, 245-254. (doi:10.1115/1.3140666)

[2] Tokhi, M. O., Poerwanto, H., and Azad, A. K. M. (1995). Dynamic simulation of flexible manipulator systems incorporating hub inertia, payload and structural damping, *Machine Vibration*, **4**, 106-124.

[3] Tokhi, M. O. and Azad, A. K. M. (1995). Active vibration suppression of flexible manipulator systems - Closed-loop control methods, *International Journal of Active Control*, **1**, 79-107.

[4] Tokhi, M. O. and Azad, A. K. M. (1996a). Control of flexible manipulator systems, *Proceedings of IMechE-I: Journal of Systems and Control Engineering*, **210**, 113-130.

[5] Tokhi, M. O. and Azad, A. K. M. (1996b). Collocated and non-collocated feedback control of flexible manipulator systems, *Machine Vibration*, **5**, 170-178.

[6] Tse, F. S., Morse, I. E., and Hinkle, T. R. (1980). *Mechanical Vibrations Theory and Applications*, Allyn and Bacon Inc.

[7] Kourmoulis, P. K. (1990). *Parallel processing in the simulation and control of flexible beam structures*, PhD Thesis, University of Sheffield, Department of Automatic Control and Systems Engineering.

[8] Elanayar, S. V. T. and Yung, C. S., (1994). Radial basis function neural network for approximation and estimation of non-linear stochastic dynamic systems, The *IEEE Transaction on Neural Networks*, **5**(4), pp. 594-603. (doi:10.1109/72.298229)

[9] Hastings ,G. S. and Book , W. J. (1987). A Linear dynamic model for flexible robotic manipulator, *IEEE Control Systems Magazine*, **7**, pp. 61-64. (doi:10.1109/MCS.1987.1105233)

[10] Korolov, V. V. and Chen,Y. H. (1989). Controller design robust to frequency variation in a one-link flexible robot arm, *Journal of Dynamic Systems, Measurement and Control*, **111**, pp. 9-14.

[11] Azad, A. K. M. (1994). *Analysis and design of control mechanisms for flexible manipulator systems*, PhD Thesis, University of Sheffield, Department of Automatic Control and Systems Engineering.

[12] Kristinsson, K. and Dumont, G. (1992). System identification and control using genetic algorithms, The *IEEE Transactions on Systems, Man and Cybernetics*, **22**(5), pp. 1033-1046. (doi:10.1109/21.179842)

[13] Hossain, M. A., Tokhi, M. O., Chipperfield, A. J., Fonseca, C. M., and Dakev, N. V. (1995). Adaptive active vibration control using genetic algorithms. The *1st IEE/IEEE International Conference on GAs in Engineering Systems: Innovations and Applications*, Sheffield, UK, pp. 175-180.

[14] Ljung, L. and Sj Ö berg, J. (1992). A system identification perspective on neural networks. *Neural Networks for Signal Processing II.-Proceedings. of the IEEE-SP Workshop,* Helsingoer, Denmark, pp. 423-435.

[15] Luo, F-L. and Unbehauen, R. (1997). *Applied neural networks for signal processing*. Cambridge University Press, Cambridge, New York.

[16] Sze, T. L. (1995). *System identification using radial basis neural networks*, PhD thesis, Department of Automatic Control and Systems Engineering, The University of Sheffield, Sheffield, UK.

[17] Shaheed, M. H. (2005). Feedforward Neural Network based Nonlinear Dynamic Modelling of a TRMS using RPROP Algorithm, *Aircraft Engineering and Aerospace Technology,***77** (1), pp. 13-22. (doi:10.1108/00022660510576000)

[18] Chipperfield, A. J. and Fleming, P. J. (1994). *Parallel Genetic Algorithms: A Survey*, Research report no. 518, Department of Automatic Control and Systems Engineering, The University of Sheffield, UK.

[19] Billings, S. A. and Voon, W. S. F. (1986). Correlation based model validity tests for non-linear systems, *International Journal of Control*, **15**, (6), pp. 601-615.

[20] The Mathworks Inc, (2002). *MATLAB Guide Users Manual*, The Mathworks Inc. Natwick.

[21] The Mathworks Inc, (2003). *SIMULINK, Users Guide,* The Mathworks Inc. Natwick.

[22] Marchand, P. and T. Holland (2003). *Graphics and GUI's with MATLAB*, 3rd edition, CRC Press.

AUTHORS

Abul K. M. Azad is an Associate Professor with the Technology Department of Northern Illinois University. He obtained a Ph.D. from the University of Sheffield (UK) in 1994. His research interests include mechatronic systems and structural control, remote laboratory, adaptive/intelligent control, mobile robotics, and educational research. In these areas, Dr. Azad has over 94 referred journal and conference papers, one edited book, and few book chapters. So far, he has attracted around $1.5M of research and development grants from various national and international funding agencies. He is active with various professional organizations along with editorial board member for a number of technical journals. Dr. Azad is involved with proposal review activities for NSF (USA), NRF (UAE), and European Commission (5th and 6th research framework). He is a member for ISO standardization committees for robots in personal care and service robots, and a program evaluator for the ABET (USA). He is a senior member of IEEE and ISA and of ASEE and IET. (azad@ceet.niu.edu)

Osman Tokhi obtained his BSc (Electrical Engineering) from Kabul University (Afghanistan) in 1978 and PhD from Heriot-Watt University (UK) in 1988. He has worked as a lecturer and Senior Lecturer at Kabul University, Glasgow College of Technology (UK) and the University of Sheffield (UK) and as a sound engineer in industry. He is currently employed as a Reader in the Department of Automatic Control and Systems Engineering, The University of Sheffield (UK). He has over 300 publications in print, including textbooks, journal and conference papers. His main research interests include active control of noise and vibration, adaptive/intelligent and soft computing techniques for modeling and control of dynamic systems, high-performance computing for real-time signal processing and control, and biomedical applications of robotics and control.

Hasan Shaheed has been a Lecturer with the Department of Engineering at Queen Mary, University of London (UK), since December 2000. He completed his PhD from the Department of Automatic Control and Systems Engineering, University of Sheffield (UK) on intelligent control and real-time simulation of flexible manipulators. His research interest includes lector-mechanical and aerospace systems, robotic systems, feed-forward feedback control of robotic manipulators, model-based and AI modeling techniques, computer interfacing and data acquisition, real-time signal processing, and parallel processing. He has published over 30 refereed journal and conference papers. He has co-authored an advanced text book and also has co-authored four book chapters of another book. He is a member of the IEE and the IEEE.

Parts of this work were supported through various grants from Northern Illinois University, USA. Submitted 1st May 2009. Published as resubmitted by the authors on 8 November 2009.