I J C T A, 8(2), 2015, pp. 715-731

System Identification Based on Heuristic Approaches

Medhat Awadalla, Dawood Al-Abri, Ali Al-Lawati and Samir Al-Busaidi

Abstract: This paper presents an investigation of the development of system identification using intelligent algorithms. A simulation platform of a flexible beam vibration using finite difference (FD) method is accomplished to demonstrate the capabilities of the identification algorithms. Three heuristic approaches for system identification are explored and evaluated. These identification approaches are, Adaptive Neuro Fuzzy Inference System (ANFIS) model, Bees Algorithm (BA) and Particle Swarm Optimization, PSO. The above approaches are used to estimate a linear discrete second order model for the flexible beam vibration. The model is implemented, tested and validated to demonstrate the merits of the algorithms for system identification. Finally, a qualitative comparison have been accomplished to address the system performance in terms of error convergence of the proposed approaches. The achieved results of intensive simulated experiments show that PSO outperforms the other approaches.

Keywords: System identification, adaptive control, intelligent identification, ANFIS, Bees Algorithm, PSO.

1. INTRODUCTION

This paper presents an investigation of the development of a discrete time model based on the observation of the input and output signals. Such models can be used for control system design, adaptive guidance or fault detection (Fei 2007). Parameter estimation, in turn system identification is a common criterion for control system, in particular for sensitive or adaptive control system design. In fact, a closed loop control system may be unstable or exhibit unacceptable transient response characteristics if the estimated parameters used in the system model for controller design do not coincide with the actual process parameters. Therefore, accurate and reliable parameters estimation technique is critical for the design and development of high-performance control systems in which the estimated parameters are often used in the field orientation, motion control, self-sensing, and other advanced algorithms.

A flexible beam system in transverse vibration is considered in this paper. Such as system has an infinite number of modes, although in most cases the lower modes are the dominant ones requiring attention. The unwanted vibrations in the structure are assumed to be the result of a single point disturbance of broadband nature. First-order central finite difference (FD) methods are used to study the behaviour of the beam and develop as suitable test and verification platform. An AVC system is designed utilizing a single input single output control structure to yield optimum cancellation of broadband vibration at a set of observation points along the beam. The controller design relations are formulated such as to allow on-line design and implementation and thus, yield an adaptive control algorithm (Long Zhao and Jing Liu 2012). It is reported earlier that the the conventional on-line system identification schemes are in essence local search techniques (Hashim *et al.* 2006). These techniques often fail in the search for the global optimum if the search apace is not differentiable or linear in the parameters. To overcome this limitation, this investigation employed GAs and ANFIS to identify characteristics of the AVC system. The evolutionary GAs and the ANFIS algorithm of the MATLAB tool boxes are used to estimate the controller characteristics, where the controller is designed based on the plant model. This is realized by minimizing the prediction error of the actual plant output and the model output. The flexible beam system mentioned above is considered as the

* Electrical and Computer Engineering Department, SQU, Oman

plant model. An AVC system is designed for optimum cancellation of broadband vibration along the beam. The AVC algorithm is designed, implemented and tested using GAs and ANFIS algorithm. The performances of the both algorithms in implementing AVC system are assessed in the suppression of vibration along the beam. These are presented and discussed through a set of experiments.

The main objective of this paper is to identify a linear discrete second order model using GAs, ANFIS, BA and PSO. A simulation platform of a flexible beam system in transverse vibration using FD method (Long Zhao and Jing Liu 2012) is considered to demonstrate the capabilities of the algorithms for system identification. The proposed second order model is implemented using the GAs, ANFIS, BA and PSO. It is then tested and validated for system identification within the simulation framework of a flexible beam system. Finally, a performance comparative of the four algorithms are presented and discussed to demonstrate the capabilities of the algorithm in implementing system identification. In next sections, all these algorithms will be presented.

2. INTELLIGENT IDENTIFICATION ALGORITHMS

The conventional system identification schemes are in essence local search techniques. These techniques often fail in the search for the global optimum if the search space is not differentiable or linear in the parameters. On the other hand, these techniques do not iterate more than once on each datum received. An alternative strategies using artificial intelligence algorithms could provide better solution. To achieve this goal, five most commonly used intelligence algorithms are used to demonstrate the capabilities.

2.1. Feed-Forward Neuro-fuzzy Technique

One major disadvantage of fuzzy approaches is that there are no clear guidelines as to how to fine-tune the fuzzy membership functions. However, learning techniques are being developed that can help in this process.

Updating the knowledge base affects the current rules and hence the system outputs. Therefore, a neuro-fuzzy technique has been proposed to fine-tune these rules and minimise the total error between the desired output and the fuzzy controller output.

Mamdani-model-based fuzzy neural networks (FNNs) represent more transparent neurofuzzy systems compared with TS-model-based FNNs (Shankir, 2001). The reason is that the rule base of the Mamdani-model is more understandable to human users. Also, it is more general in terms of how its rule base is created, because the latter can be constructed using human experience and numerical data. However, a disadvantage of this model is that it does not allow easy mathematical analysis due to the logical nature of its inference functions, e.g. the logic min/max functions. Also, it does not allow the simple application of BP as one of the most powerful learning algorithms, due to the non-differentiable min/max functions employed.

In this chapter, a Mamdani-model-based FNN with Differentiable Activation functions (DA-FNN) is described. A differentiable alternative to the logic min and logic max functions termed softmin and softmax (Shankir, 2001) are presented. These two differentiable functions (softmin and softmax) are employed instead of the two non-differentiable functions (logic min and logic max) to implement the decision-making mechanism of DA-FNN. Using these differentiable functions allows the effective application of BP for the parameter learning of DA-FNN.

Figure 1(a) presents the structure of the proposed neuro-fuzzy system. The structure is a six-layer feed-forward connectionist representation of a Mamdani-model-based fuzzy logic system (FLS) (Mamdani, 1974). In general, a node in any layer has some finite "fan-in" of connections represented by weight values from other nodes and a "fan-out" of connections to other nodes (see Figure 1(b)). Associated with the fan-in of a node is an aggregation function, f, that serves to combine information, activation, or evidence from other nodes. Using the same notations as in (Lin and Lee, 1992), the function that provides the net input to such a node is written as follows:

$$\text{net input} = f^k\left(u_1^k, u_2^k, \ldots\ldots\ldots, u_p^k; w_1^k, w_2^k, \ldots\ldots\ldots, w_p^k\right) \quad (1)$$

where p is the number of fan-ins of the node, w is the link weight associated with each fan-in, u is an output of a node in the preceding layer associated with the fan-in and the superscript indicates the layer number. A second action of each node is to output an activation value as a function of its net input,

$$\text{output} = o_i^k = a^k\left(f^k\right) \quad (2)$$

where a^k denotes the activation function in layer k. The functions of the nodes in each of the six layers of the proposed structure are described next.

Layer 1: Nodes in Layer 1 are input nodes that represent input linguistic variables. Layer one contains N nodes, each of which receives a crisp input vector $\mathbf{x} = (x_1, \ldots, x_N)$. The nodes in this layer simply transmit input values to the next layer directly. That is,

$$f_i^1 = u_i^1 = x_i \quad \text{and} \quad a_i^1 = f_i^1 \quad (3)$$

The link weights in Layer 1 are fixed at unity.

Layer 2: Nodes in Layer 2 are input term nodes which act as membership functions. An input linguistic variable x in a universe of discourse U is characterised by $\mathbf{T(x)} = \{T_x^1, T_x^2, \ldots, T_x^N\}$ and $\mathbf{M(x)} = \{M_x^1, M_x^2, \ldots, M_x^N\}$, where T(x) is the term set of x; that is the set of names, e.g. (small, medium, large), of the linguistic values of x and M(x) is the membership function, e.g. (triangular, trapezoidal, bell-shaped), defined on a universe U, bell-shaped function is chosen because it is differentiable function. The function of each node j in a term set i is to calculate the degree of membership of input xi with respect to the membership function $M_{x_i}^j, j = 1,2,\ldots,N_i$, associated with the term set T(xi) according to the following bell-shaped function:

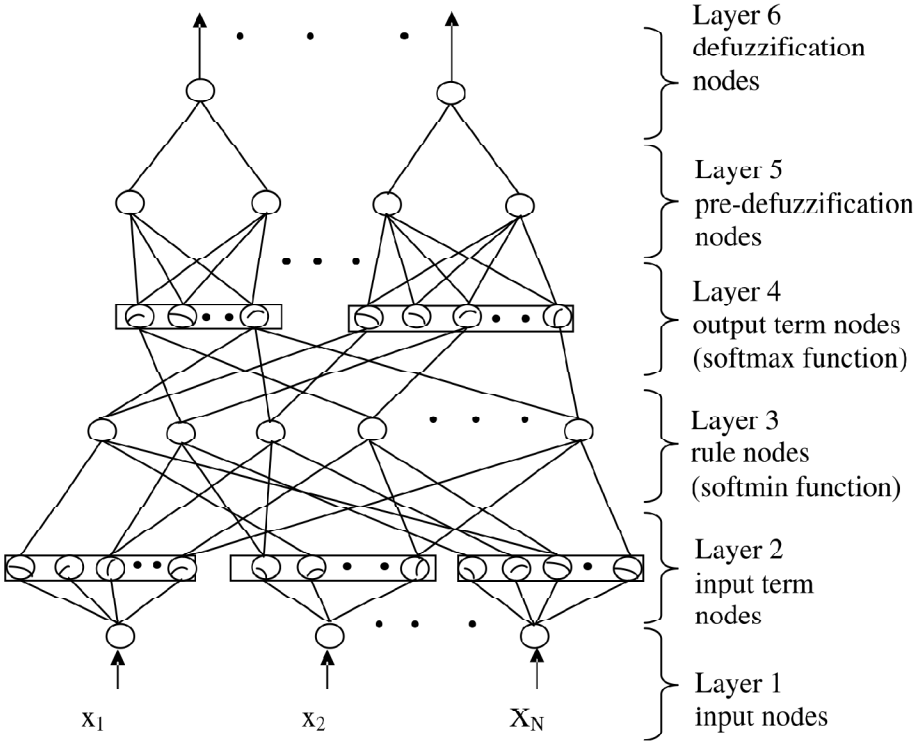

Figure 1(a): Structure of the proposed neuro-fuzzy system

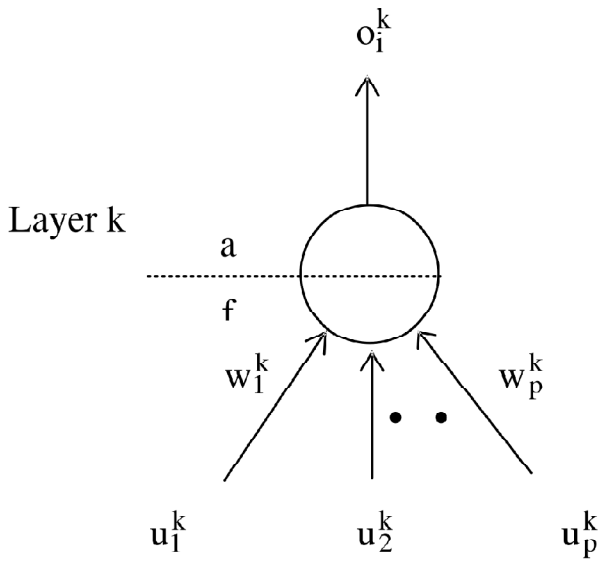

Figure 1(b): Basic structure of a node in the network

$$f_{ij}^2 = M_{x_i}^j(mij, \sigma ij) = \frac{\left((w_{ij}^2 * a_i^1) - m_{ij}\right)^2}{\sigma_{ij}^2} \quad \text{and} \quad a_i^2 = e^{-f_{ij}^2} \tag{4}$$

where m_{ij} and σ_{ij} are, respectively, the centre (or mean) and the width (or variance) of the bell-shaped function of the j^{th} term of the i^{th} input linguistic variable x_i.

Layer 3: The nodes in Layer 3 are rule nodes, where each node associates one term node from each term set to form a condition part of one fuzzy rule. In this structure, the softmin function (Berenji and Khedkar, 1992) and its complement softmax function (Shankir, 2001) are used.

$$softmin\left(a_i, i = 1,2,...,n\right) = \frac{\sum_{i=1}^{n} a_i e^{-k a_i}}{\sum_{i=1}^{n} e^{-k a_i}} \tag{5}$$

$$softmax\left(a_i, i = 1,2,...,n\right) = \overline{softmin(\overline{a_i}, i = 1,.2,..,n)} = \left[1 - \frac{\sum_{i=1}^{n} \overline{a_i} e^{-k\overline{a_i}}}{\sum_{i=1}^{n} e^{-k\overline{a_i}}}\right] \tag{6}$$

where $a_i = \mu$, $\overline{a_i} = 1 - a_i$ and the parameter k controls the "hardness" of the softmin function.

Therefore, the function of the r^{th} rule node using softmin can be written as follows:

$$f_r^3 = softmin(u_1^3, u_2^3, ..., u_N^3) \quad and \quad a_r^3 = f_r^3 \tag{7}$$

where r = 1,…, R, and R is the number of rules or rule nodes in layer three. However, in this layer, there are no link weights to be adjusted because all the link weights are fixed at unity.

Layer 4: The nodes in this Layer are output term nodes which act as membership functions to represent the output terms of the respective L linguistic output variables. The nodes in Layer 4 should integrate the fired rules that have the same consequent. The softmax function is used to perform the integration. Therefore, the function of each term node j in the output term set i can be written as follows:

$$f_{ij}^{4} = \underset{m}{softmax}\left(a_{m}^{3}, m = 1,...,p\right) \quad and \quad a_{ij}^{4} = f_{ij}^{4} \tag{8}$$

where p is the number of rules sharing the same consequent (the same output term node). Hence, the link weights in Layer 4 are fixed at unity.

Layer 5: The number of nodes in Layer 5 is 2L, where L is the number of output variables, i.e. there are two nodes for each output variable. The function of these two nodes is to calculate the denominator and the numerator of a quasi-Centre Of Area (COA) defuzzification value for each output variable. The functions of the two nodes of the ith output variable are:

$$f_{ni}^{5} = \sum_{j} a_{ij}^{4} * m_{ij} * \sigma_{ij} \quad and \quad a_{ni}^{5} = f_{ni}^{5} \tag{9}$$

$$f_{di}^{5} = \sum_{j} a_{ij}^{4} * \sigma_{ij} \quad and \quad a_{di}^{5} = f_{di}^{5} \tag{10}$$

where f_{ni}^{5} and f_{di}^{5} are, respectively, the node functions of the numerator and the denominator nodes of the i^{th} output variable.

Layer 6: The nodes in Layer 6 are defuzzification nodes. The number of nodes in this Layer equals the number of output linguistic variables. The function of the i^{th} node corresponding to the i^{th} output variable can be written as follows:

$$f_{i}^{6} = \frac{w_{ni}^{6} * a_{ni}^{5}}{w_{di}^{6} * a_{di}^{5}} \quad and \quad a_{i}^{6} = f_{i}^{6} \quad and \quad y_{i} = a_{i}^{6} \tag{11}$$

where w_{ni}^{6} and w_{di}^{6} are Layer 6 link weights associated with each output variable node.

In order to build a neuro-fuzzy system based on the above description, three main steps have to be considered. The first step is to specify the input and output variables of the network. The second step is to divide the input-output universes into a suitable number of partitions (fuzzy sets) and to specify a membership function for each partition. A linguistic term has to be assigned to each membership function and the parameters of the membership function (centre and width) have to be specified initially. The third step is to generate fuzzy rules to perform the input-output mapping of the FLS. After the construction of the network, a parameter learning phase has to be conducted. The algorithm for that phase is explained next.

5.5.1. Parameter Learning Algorithm

Following the construction phase, the network then enters the parameter learning phase to adjust its free parameters. The adjustable free parameters were selected to be the centres (m_{ijs}) and widths (s_{ijs}) of the term nodes in Layer 4 as well as the link weights in Layers 2 and 6. A supervised learning technique is employed along with the back propagation (BP) learning algorithm (Berenji and Khedkar, 1992) to tune these parameters. The problem can be stated as: 'Given n input patterns $x_i(t)$, i = 1,...., n, m desired output patterns $y_i(t)$, i = 1,....., m, the fuzzy partitions, and the fuzzy rule base, adjust the free parameters of the network optimally'. In the parameter learning phase, the goal is to minimise the following error function:

$$E = \frac{1}{2}\left(\mathrm{y}(t) - y_{net}(t)\right)^2 \tag{12}$$

where $y(t)$ is the desired output, and $y_{net}(t)$ is the current network output. For each training data set, starting at the input nodes, a forward pass is made to compute the activity levels of all the nodes in the network. Then, starting at the output nodes, a backward pass is followed to compute the rate of change of the error function with respect to the adjustable free parameters for all the hidden nodes. Assuming that w is an adjustable free parameter in a node, then the general learning rule can be written as follows:

$$\Delta w = \eta \left(-\frac{\partial E}{\partial w}\right) \tag{13}$$

$$w(t+1) = w(t) + \Delta w \tag{14}$$

where η is the learning rate. Using the chain rule, the partial derivative can be defined thus:

$$\frac{\partial E}{\partial w} = \frac{\partial E}{\partial(\text{net} - \text{input})}\frac{\partial(\text{net} - \text{input})}{\partial w} = \frac{\partial E}{\partial f}\frac{\partial f}{\partial w} = \frac{\partial E}{\partial a}\frac{\partial a}{\partial f}\frac{\partial f}{\partial w} \tag{15}$$

The calculation of the back-propagated errors as well as the updating of the free parameters can be described starting at the output nodes as follows.

Layer 6: Using Equation (15) and Equation (11), the adaptive rule for the Layer 6 weights is derived below:

$$\frac{\partial E}{\partial w_{ni}} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial w_{in}} = -\left(y(t) - y_{net}(t)\right) * \frac{a_{ni}^5}{w_{di} a_{di}^5} \tag{16a}$$

$$w_{ni}(t+1) = w_{ni}(t) + \eta_6\left(-\frac{\partial E}{\partial w_{ni}}\right) \tag{16b}$$

$$\frac{\partial E}{\partial w_{di}} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial w_{di}} = -\left(y(t) - y_{net}(t)\right) * -\frac{w_{ni} a_{ni}^5}{(w_{di})^2 a_{di}^5} \tag{17a}$$

$$w_{di}(t+1) = w_{di}(t) + \eta_6\left(-\frac{\partial E}{\partial w_{di}}\right) \tag{17b}$$

where η_6 is the learning rate of the link weights in Layer 6. The propagated errors from Layer 6 to the numerator and the denominator nodes in Layer 5 are derived as follows:

$$\delta_{ni}^6 = \frac{\partial E}{\partial a_{ni}^5} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial a_{ni}^5} = -\left(y(t) - y_{net}(t)\right) * \frac{w_{ni}}{w_{di} a_{di}^5} \tag{18a}$$

$$\delta_{di}^6 = \frac{\partial E}{\partial a_{di}^5} = \frac{\partial E}{\partial a_i^6} * \frac{\partial a_i^6}{\partial f_i^6} * \frac{\partial f_i^6}{\partial a_{di}^5} = -\left(y(t) - y_{net}(t)\right) * -\frac{w_{ni} a_{ni}^5}{w_{di}\left(a_{di}^5\right)^2} \tag{18b}$$

Layer 5: An adjustment is required for the link weights w_{nij}^5 which represent the centres mij of the output membership functions. Also, an adjustment is required for the free parameter sij that represents the width of the output membership functions. Consequently, using Equation (9) and Equation (15), the adaptive rule to tune the free parameters in Layer 5 is derived. First, the adaptive rule to tune the centres of the output membership functions can be obtained as follows:

$$\frac{\partial E}{\partial m_{ij}} = \frac{\partial E}{\partial a_{ni}^5} * \frac{\partial a_{ni}^5}{\partial f_{ni}^5} * \frac{\partial f_{ni}^5}{\partial m_{ij}} = \delta_{ni}^6 * \sigma_{ij} * a_{ij}^4 \tag{19a}$$

$$m_{ij}(t+1) = m_{ij}(t) + \eta_5 \left(-\frac{\partial E}{\partial m_{ij}} \right) \tag{19b}$$

Second, the adaptive rule to tune the widths of the output membership functions is given by:

$$\frac{\partial E}{\partial \sigma_{ij}} = \frac{\partial E}{\partial a_{ni}^5} * \frac{\partial a_{ni}^5}{\partial f_{ni}^5} * \frac{\partial f_{ni}^5}{\partial \sigma_{ij}} = \delta_{ni}^6 * m_{ij} * a_{ij}^4 \tag{20a}$$

$$\sigma_{ij}(t+1) = \sigma_{ij}(t) + \eta_5 \left(-\frac{\partial E}{\partial \sigma_{ij}} \right) \tag{20b}$$

where η_5 is the learning rate of the adjustable parameters σ_{ij} and m_{ij} in Layer 5. The propagated error from Layer 5 to the j^{th} node in the i^{th} term set in Layer 4 is:

$$\delta_{ij}^5 = \left(\frac{\partial E}{\partial a_{ni}^5} * \frac{\partial a_{ni}^5}{\partial f_{ni}^5} * \frac{\partial f_{ni}^5}{\partial a_{ij}^4} \right) + \left(\frac{\partial E}{\partial a_{di}^5} * \frac{\partial a_{di}^5}{\partial f_{di}^5} * \frac{\partial f_{di}^5}{\partial a_{ij}^4} \right) = \left(\delta_{ni}^6 * m_{ij} * \sigma_{ij} \right) + \left(\delta_{di}^6 * \sigma_{ij} \right) \tag{21}$$

Layer 4: No adjustment is required for the link weights of Layer 4. Only the error signals δ_r^4 are required to be calculated and propagated to a rule node r in Layer 3. Each one of these error signals is a summation of L propagated error signals δ_{ri}^4, one error signal from a specific node j of each term set i, where i = 1,…., L and L is the number of output variables (or term sets). Using Equation (15), the error signal δ_r^4 is then:

$$\delta_r^4 = \sum_i \delta_{ri}^4 = \sum_i \left(\delta_{ij}^5 * \frac{\partial a_{ij}^4}{\partial f_{ij}^4} * \frac{\partial f_{ij}^4}{\partial a_r^3} \right) \tag{22}$$

From Equations (5.8) and (5.5)

$$\frac{\partial a_{ij}^4}{\partial f_{ij}^4} = 1 \tag{23}$$

and

$$\frac{\partial f_{ij}^{4}}{\partial a_r^3} = \frac{e^{-k\bar{a}_r^3}\left[\left(1-k\bar{a}_r^3\right)\sum_{m=1}^{p} e^{-k\left(\bar{u}_{ijm}^4\right)} + k\sum_{m=1}^{p}\bar{u}_{ijm}^4 e^{-k\left(\bar{u}_{ijm}^4\right)}\right]}{\left(\sum_{m=1}^{p} e^{-k\left(\bar{u}_{ijm}^4\right)}\right)^2}, \tag{24}$$

if the j[th] term node at the i[th] term set in Layer 4 is connected to the r[th] rule node in Layer 3. Otherwise,

$$\frac{\partial f_{ij}^{4}}{\partial a_r^3} = 0 \tag{25}$$

where p is the number of rules sharing the same j[th] output term node, and $\bar{u}_{ijm}^4$ is the complement of the m[th] input to the j[th] output term node at the i[th] term set in Layer 4.

Layer 3: As with Layer 4, no adjustment is required for link weights in Layer 3. Only the error signals δ_{ij}^3 are required to be calculated and propagated from the rth rule node in Layer 3 to the j[th] term node at the i[th] term set in Layer 2. Each one of these error signals is a summation of p propagated error signals δ_{ijm}^3 from Layer 3, where $m = 1,..., p$, and p is the number of rules which share the same jth term node at the same i[th] input term set in Layer 2. Using Equation (15), the error signal δ_{ij}^3 can be calculated as follows:

$$\delta_{ij}^3 = \sum_m \delta_{ijm}^3 = \sum_m \left(\delta_m^4 * \frac{\partial a_m^3}{\partial f_m^3} * \frac{\partial f_m^3}{\partial a_{ij}^2} \right) \tag{26}$$

From Equations (7) and (6)

$$\frac{\partial a_m^3}{\partial f_m^3} = 1 \tag{27}$$

and

$$\frac{\partial f_m^3}{\partial a_{ij}^2} = \frac{e^{-k a_{ij}^2}\left[\left(1-k a_{ij}^2\right)\sum_{i=1}^{N} e^{-k u_{mi}^3} + k\sum_{i=1}^{N} u_{mi}^3 e^{-k u_{mi}^3}\right]}{\left(\sum_{i=1}^{N} e^{-k u_{mi}^3}\right)^2}, \tag{28}$$

if the j[th] term node at the i[th] input term set in Layer 2 is connected to the rule node m in Layer 3; otherwise,

$$\frac{\partial f_m^3}{\partial a_{ij}^2} = 0 \tag{29}$$

where N is the number of input term sets and u_{mi}^3 is the i[th] input to the rule node m in Layer 3.

Layer 2: Using Equation (5.13) and Equation (5.4), the adaptive rule to tune the weights in Layer 2 is given by:

$$\frac{\partial E}{\partial w_{ij}^2} = \frac{\partial E}{\partial a_{ij}^2} * \frac{\partial a_{ij}^2}{\partial f_{ij}^2} * \frac{\partial f_{ij}^2}{\partial w_{ij}^2} = \delta_{ij}^3 * e f_{ij}^2 * \frac{-2 a_i^1 \left(a_i^1 w_{ij}^2 - m_{ij}\right)}{\sigma ij^2} \tag{30a}$$

$$w_{ij}^2(t+1) = w_{ij}^2(t) + \eta_2 \left(-\frac{\partial E}{\partial w_{ij}^2}\right) \tag{30b}$$

where η_2 is the learning rate of the link weights in Layer 2. The propagated error from Layer 2 to the i^{th} input node in Layer 1 is derived as:

$$\delta_i^2 = \sum_j \delta_{ij}^2 = \sum_j \left[\frac{\partial E}{\partial a_{ij}^2} * \frac{\partial a_{ij}^2}{\partial f_{ij}^2} * \frac{\partial f_{ij}^2}{\partial a_i^1}\right] = \sum_j \left[\delta_{ij}^3 * f_{ij}^2 * \frac{-2 w_{ij}^2 \left(a_i^1 w_{ij}^2 - m_{ij}\right)}{\sigma ij^2}\right] \tag{31}$$

Following the construction phase and the learning phase, an optimally tuned FLS is developed to perform a specific mapping function. This mapping function may represent a function of a dynamic system or a control function.

3.2. Bees Algorithm

The bees algorithm is an optimization algorithm inspired by the natural foraging behaviors of honey bees to find the optimal solution. It belongs to the category of "intelligent" optimization tools as it also simultaneously evaluates many points in the parameter space and converges towards the global solution and is also based on the method of minimization of the prediction error. Bees Algorithm requires a number of parameters to be set, namely: number of scout bees (n), number of sites selected out of n visited sites (m), number of best sites out of m selected sites (e), number of bees recruited for best e sites (nep), number of bees recruited for the other (m-e) selected sites (nsp), initial size of patches (ngh) which includes site and its neighbourhoods and stopping criterion. The procedure for BA can be written as follows:

1. Initialize population with random solutions.
2. Evaluate fitness of the population.
3. While (stopping criterion not met) - format new population.
4. Select sites for neighbourhood search.
5. Recruit bees for selected sites (more bees for best e sites) and evaluate fitness's.
6. Select the fittest bee from each patch.
7. Assign remaining bees to search randomly and evaluate their fitness's.
8. End While

The algorithm starts with the n scout bees being placed randomly in the search space. The fitness's of the sites visited by the scout bees are evaluated in step 2. In step 4, bees that have the highest fitness's are chosen as "selected bees" and sites visited by them are chosen for neighborhoods search. Then, in steps 5 and 6, the algorithm conducts searches in the neighborhoods of the selected sites, assigning more bees to search near to the best e sites. The bees can be chosen directly according to the fitness's associated with the sites they are visiting. Alternatively, the fitness values are used to determine the probability of the bees being selected. Searches in the neighborhood of the best e sites which represent more promising solutions are made more detailed by recruiting more bees to follow them than the other selected bees. Together with

scouting, this differential recruitment is a key operation of the Bees Algorithm. However, in step 6, for each patch only the bee with the highest fitness will be selected to form the next bee population. In nature, there is no such a restriction. This restriction is introduced here to reduce the number of points to be explored. In step 7, the remaining bees in the population are assigned randomly around the search space scouting for new potential solutions. These steps are repeated until a stopping criterion is met. At the end of each iteration, the colony will have two parts to its new population – representatives from each selected patch and other scout bees assigned to conduct random searches (Pham and Castellani 2009; Packianather et al. 2009; Pham 2006).

3.3. PSO

Sedighizadeh and Masehian (Sedighizadeh, and Masehian 2009) developed PSO algorithm simulating the behaviour of swarms in the nature, such as birds, fish, etc. In PSO, the potential solutions, called particles, fly through the problem space by following the current optimum particles. PSO has been successfully applied in many scientific areas and there are many variants of the algorithm. A survey of PSO methods and applications could be found in (Lalwani *et al.* 2013). At the beginning, a set (swarm) of random solutions (particles) is used to initialize PSO algorithm that starts iterations looking for optimal solution. During every iteration, each particle is updated by two best values. The first one is the personal best pbest that the particle has achieved so far. The second is the global best gbest obtained by any particle in the swarm. After finding these two best values, the particle updates its velocity and position according to equations (33) and (34) respectively.The typical procedure of PSO is shown as follows:

initialize the population randomly.

DO

{

For each particle.

{

Calculate fitness value If the fitness value is better than the best fitness value (pbest) in history then set current value as the new pbest.

}

Choose the particle with the best fitness value of all particles as the gbest.

For each particle.

{

Calculate new velocity:

$$Vnew= W.Vo1d +C1.R1.(pbest - X) + C2.R2.(gbest-X) \quad (33)$$

Where W is inertia constant, R1 and R2 are random values and C1 and C2 are constant values and X is particle position.

Update particle position:

$$Xnew = Xold + Vnew \quad (34)$$

}

}

Until termination criterion is met.

The random numbers R1 and R2 are generated uniformly between 0 and 1 and the constants C1 (self-knowledge factor) and C2 (social-knowledge factor) are usually in the range from 1.5 to 2.5. Finally, the

inertia factor W can be fixed or varied with a decreasing value as the algorithm proceeds or it may be restarted as in (Lalwani *et al.* 2013). A possible solution (particle) is a vector of n elements, where each element is associated to a given force. Thus, the search space size is mn There are k particles in the swarm that form swarm (population) size; these particles are initialized randomly.

4. THE FLEXIBLE BEAM SYSTEM

Consider a cantilever beam as a plant model of length *L*, shown in figure , fixed at one end and free at another, with a force *U*(*x*, *t*) applied at a distance *x* from its fixed (clamped) end at time *t*, resulting a deflection *y*(*x*, *t*) of the beam from its stationary (fixed) position at the point where the force has been applied. The motion of the beam in transverse vibration is, thus, governed by the well known fourth-order partial differential equation (PDE) (Madkour 2007).

$$\mu^2 \frac{\partial^4 y(x,t)}{\partial x^4} + \frac{\partial^2 y(x,t)}{\partial t^2} = \frac{1}{m} U(x,t) \tag{35}$$

Where μ is a beam constant given by $\mu^2 = \frac{EI}{\rho A}$, with α, A, I and E representing the mass density, cross-sectional area, moment of inertia of the beam and the Young modulus respectively, and m is the mass of the beam. The corresponding boundary conditions at the fixed and free ends of the beam are given by

$$\begin{aligned} &y(0,t) = 0 \qquad and \qquad \frac{\partial y(0,t)}{\partial x} = 0 \\ &\frac{\partial^2 y(L,t)}{\partial x^2} = 0 \quad and \quad \frac{\partial^3 y(L,t)}{\partial x^3} = 0 \end{aligned} \tag{36}$$

Note that the model thus utilised incorporates no damping. To construct a suitable platform for test, a method of obtaining numerical solution of the PDE in equation (35) is required. This can be achieved by using the finite difference (FD) method. This involves a discrimination of the beam into a finite number of equal-length sections (segments), each of length Δ*n*, and considering the beam motion (deflection) for the end of each section at equally-spaced time steps of duration Dt. Thus, first-order central FD methods is used to approximate the partial derivative terms in equations (16) and (17) yields.

$$Y_{j+1} = -Y_{j-1} - \lambda^2 S Y_j + (\Delta t)^2 U(x,t) \frac{1}{m} \tag{37}$$

where, Y_k ($k = j-1,\ j,\ j+1$) is an n×1 matrix representing the deflection of grid-points 1 to n of the beam at time step *k*, S is a matrix, given in terms of characteristics of the beam and the discrimination steps Δ*t* and Δ*x*, and $\lambda^2 = (\Delta t)^2 (\Delta x)^{-4} \mu^2$. Equation (24) is the required relation for the simulation algorithm, characterising the behaviour of the cantilever beam system, which can be implemented on a digital computer easily. It has been shown that a necessary and sufficient condition for stability satisfying this convergence requirement is given by $0 < \lambda^2 \leq 0.25$ (Tavakolpour *et al.* 2009).

5. IMPLEMENTATION AND RESULTS

A cantilever beam in transverse vibration of length L = 0.635 meter, mass m = 0.037 kg, was considered as plant for investigation. The beam was discretised into 19 small segments. To allow dominant modes of vibration of the beam to be excited, a step disturbance force (0.1N) of finite duration was applied to a suitable node of the beam. The input and output samples of the plant was collected from two suitable nodes

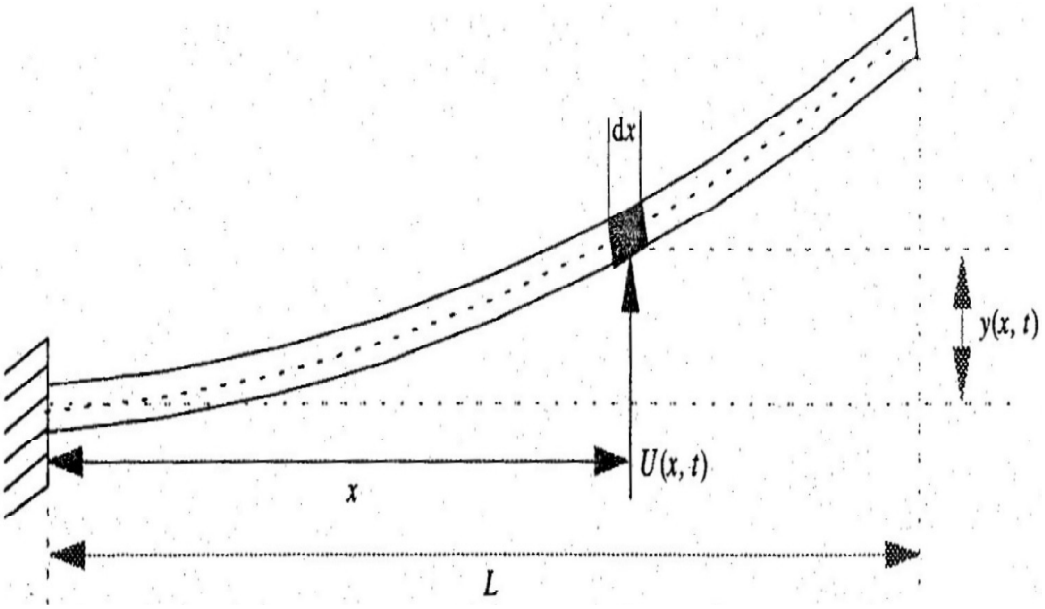

Figure 2: Schematic diagram of the cantilever beam system

of the beam. Moreover, sample period was selected as $\Delta t = 0.3$ ms which is sufficient to cover all the resonance modes of vibration of the beam (Tavakolpour 2009).

A linear discrete second order model was estimated using the ANFIS, BA and PSO.

$$Y(z) = \frac{1 + b_1(z^{-1}) + b_2(z^{-2})}{1 + a_1(z^{-1}) + a_2(z^{-2})} U(z) \tag{38}$$

Investigations were carried out using Fuzzy Logic Toolbox Version 2.2.4 for ANFIS. The system identification algorithms were carried out for about 5s (16,700 iterations) using the linear discrete second order model (equation 38) with grid points 16 and 20 as an input and output samples of the plant respectively with a set of input output data simulated using equation 37.

BA and PSO are used to estimate the parameters of the model of equation (38). On the other hand, ANFIS is used to estimate the equivalent model using plant input and corresponding output. Table 1 shows the summary of the error convergence performances of the algorithms. The error has been calculated based on the differences between absolute value of the original and the estimated signal. On the other hand, the execution time of the algorithms was measured for 16,700 iterations. It is noted that error convergences for all the algorithms are in similar level. It is also noted that PSO perform better as compared to ANFIS and BA offers the best performance among the three algorithms.

Table 1
Performance evaluation of proposed

Algorithm	*Error*
ANFIS	0.6559
BA	0.6079
PSO	0.6032

Figure 4 shows the time domain performance of ANFIS, BA, and PSO where the solid signal represents actual output and doted one represents the estimated output of the model. It is observed that a significant error convergence leads almost overlapping of the two signals in each case. It is also noted that the PSO

offers similar level of performance for error convergence as compared to the other two algorithms. Corresponding auto-power spectral density is shown in Figure 5, which further demonstrated the similarity and level of error convergence. As shown in Figure 4, the solid signal in Figure 5 represents actual output and doted one represents the estimated output of the model.

6. COMPARATIVE ANALYSIS FOR ACTIVE VIBRATION CONTROL PERFORMANCE

This paper presents the experimental results of Active Vibration Control Systems using the various intelligent algorithms. The results are discussed to demonstrate the capabilities of these algorithms.

6.1. Impact of Identification Algorithms in Control System Design

As discussed earlier, the ANFIS, BA, and PSO algorithms were used to estimate the parameters of the AVC system based on the input and corresponding output of the plant model, and the system models were developed based on the input and the cancellation signal required for destructive interference at the control

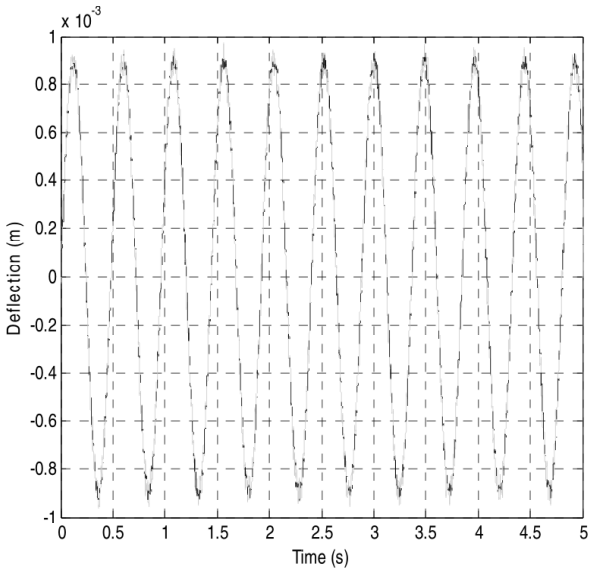

Figure 4a: Performance evaluation of ANFIS

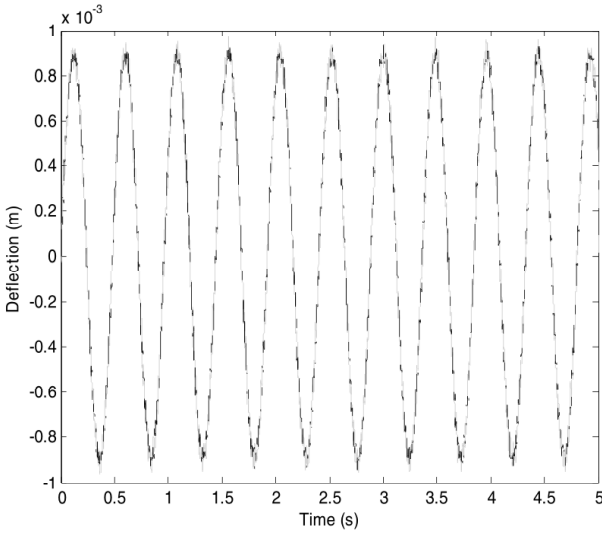

Figure 4b: Performance evaluation of BA

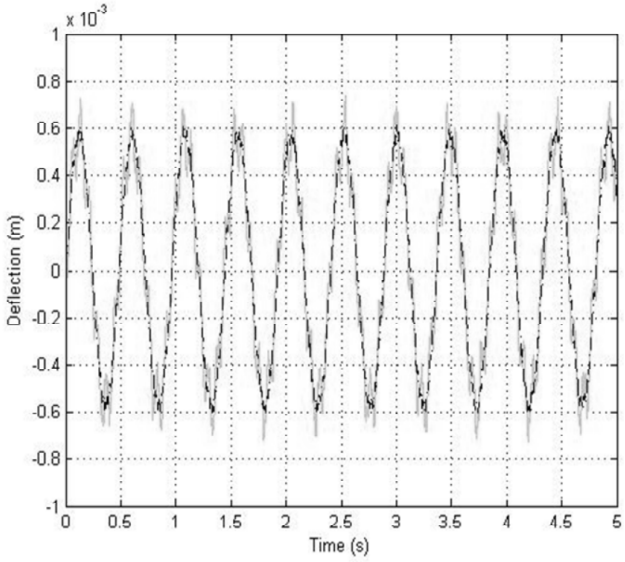

Figure 4a: Performance evaluation of PSO

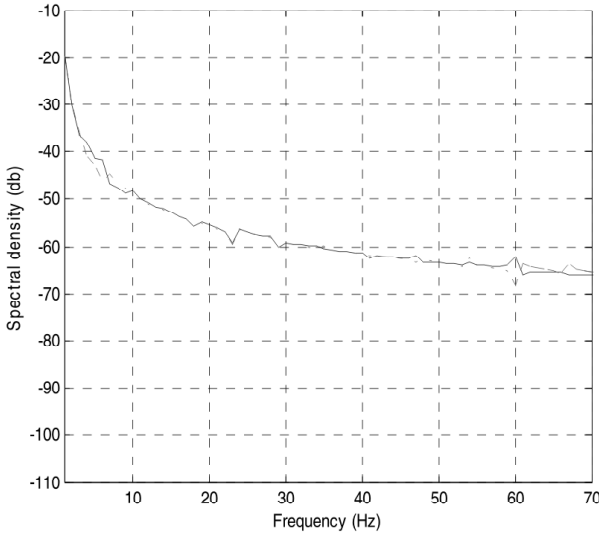

Figure 5a: Performance evaluation of ANFIS in auto-power spectral density algorithm

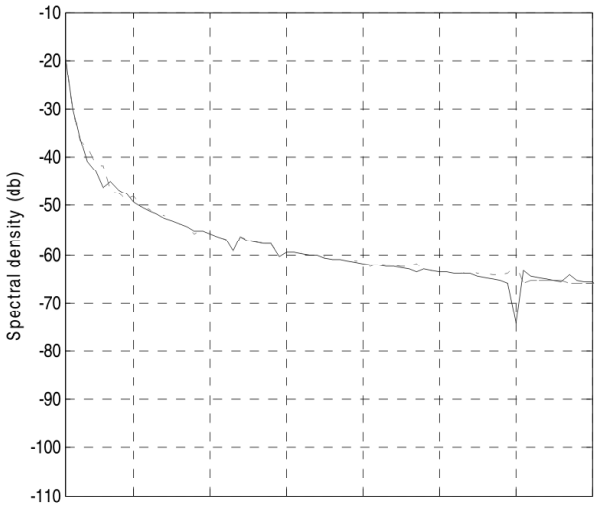

Figure 5b: Performance of BA in auto-power spectral density

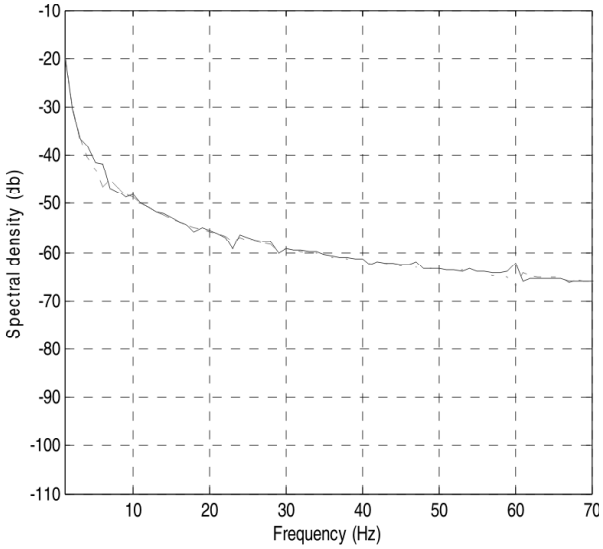

Figure 5a: Performance of PSO in auto-power spectral density

point. It is worth noting that the output signal with 180° phase shift is considered as cancellation signal. The locations of the input sensor and the output actuator were selected for the best performance based on the previous investigation as discussed in the system identification section. The AVC system was then tested and validated through a set of experiments.

6.2. Beam fluctuation at the end point before and after cancellation by AVC using the intelligent algorithms

Figure 6.1 shows the time domain response at the end point of the beam before and after cancellation. Corresponding vibration after cancellation using different algorithms are shown in 6(a) ANFIS, (e) BA and (f) PSO, respectively.

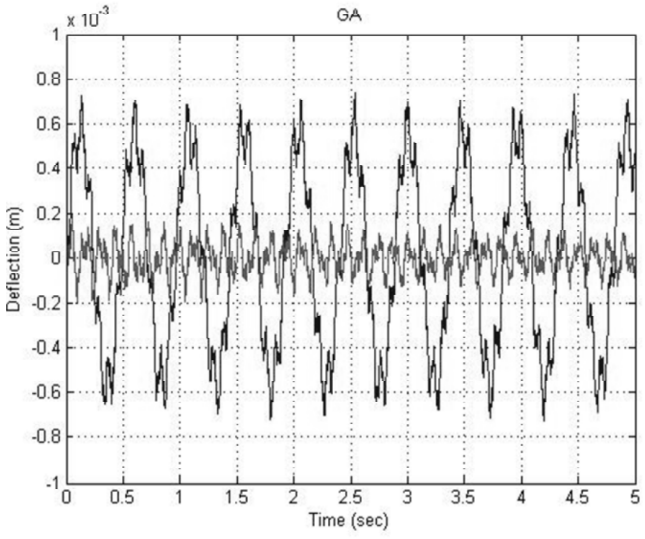

Figure 6.1a: Beam fluctuation before and after force cancellation using ANFIS

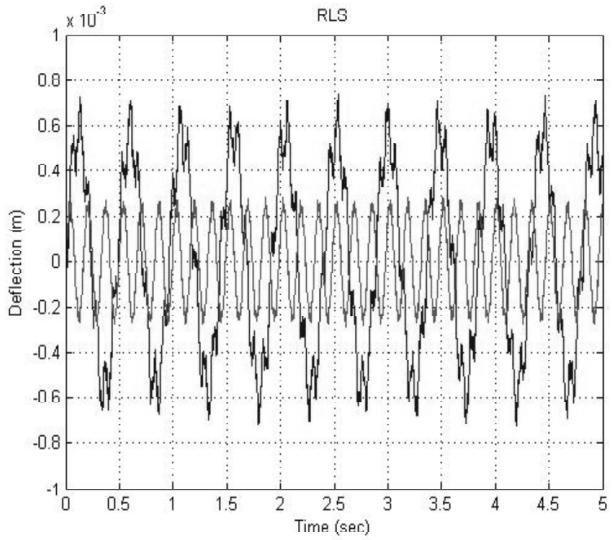

Figure 6.1b: Beam fluctuation before and after force cancellation using BA

It is noted in the identification section that PSO offers best convergence among all the algorithms. This, in turn, helps to identify more accurate controller parameters and results in best performance among all the algorithms in implementing the AVC system.

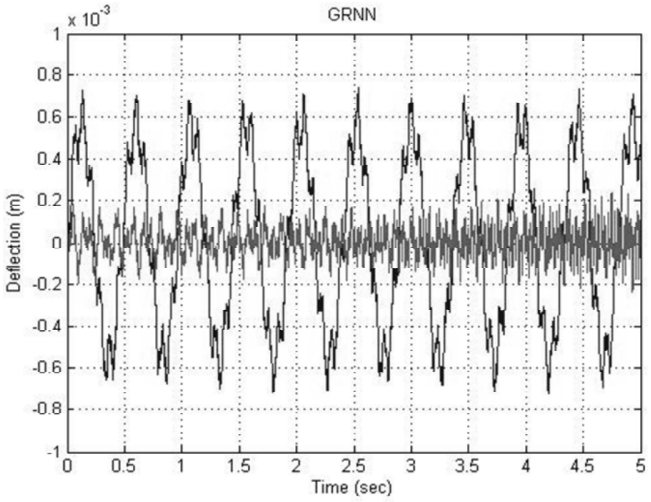

Figure 6.1c: Beam fluctuation before and after force cancellation using PSO

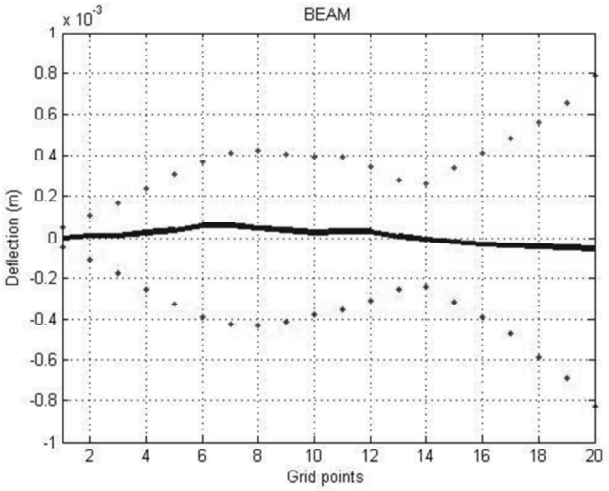

Figure 6.2a: Beam fluctuation range after cancellation using (ANFIS)

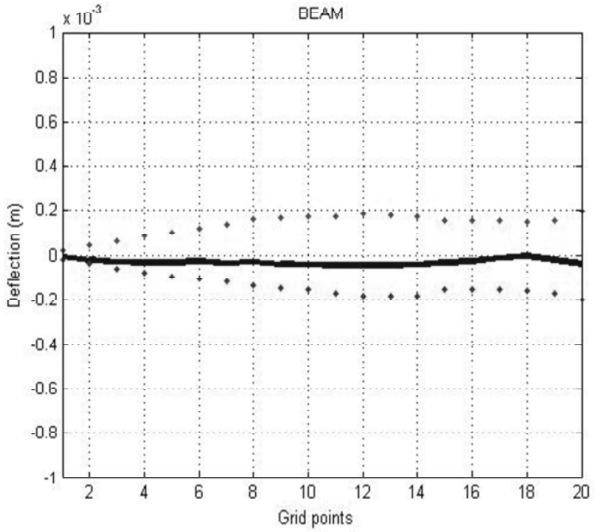

Figure 6.2b: Beam fluctuation range after cancellation using (BA)

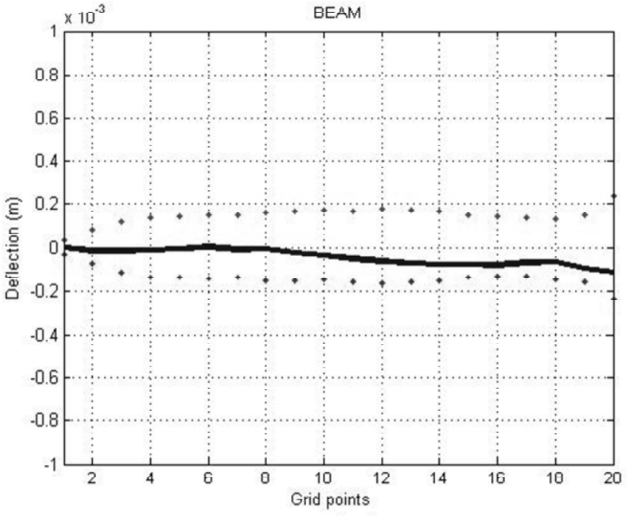

Figure 6.2c: Beam fluctuation before and after force cancellation using PSO

6.3. The overall range of fluctuations for the beam in time domain along the grid points in 2D form

Figure 6.4 demonstrates the overall range of fluctuations for the beam in time domain along the grid points in 2D form. The dotted line in the Figure depicts the maximum and minimum amplitude of the deflection along the grid points after cancellation and the solid line depicts the amplitude of deflection of each algorithm at a random sampling period. It is further evidence that overall performance of the BGA and BA are better as compared to the other algorithms.

The experimental results and convergence analysis show that the BGA and BA performed consistently better than the other algorithms. We believe the learning aspect of these two algorithms play a significant role during the simulation process.

It is noted in the identification section that PSO offers best convergence among all the algorithms. This, in turn, helps to identify more accurate controller parameters and results in best performance among all the algorithms in implementing the AVC system.

It is observed that the peak to peak amplitude at the end point before cancellation was +219.50 μm to -222.94 μm by ANFIS, +239.62 μm to -235.03 μm by BA, +193.06 μm to -199.42 μm by PAO.

6. CONCLUSION

This paper has presented the intelligent system identification of a flexible beam system for adaptive active vibration control using finite difference (FD) method is used to demonstrate the capabilities of the identification algorithms. A number of approaches and algorithms for system identification are explored and evaluated. These identification approaches are traditional Recursive Least Square (RLS) filter, Genetic Algorithms (GAs), Adaptive Neuro Fuzzy Inference System (ANFIS) model, General Regression Neural Network (GRNN), Bees Algorithm (BA) and Particle Swarm Optimization, PSO. A plant model for a flexible beam system was considered to demonstrate the merits of the algorithms. A comparative study of the heuristic algorithms has been presented and discussed through a set of experiments. PSO outperforms in terms of system convergence for the same number of iterations.

References

[1] Fei J, (2007), Adaptive sliding mode vibration control schemes for flexible structure system. Presented at the Decision and Control 2007 46th IEEE Conference, New Orleans (USA).

[2] Zhao L, Liu J. (2012), An improved adaptive filtering algorithm with applications in integrated navigation. Third International Conference on Digital Manufacturing & Automation: 182-185.

[3] Hashim S, Tokhi M, Darus I (2006), Active vibration control of flexible structures using genetic optimisation. Journal of Low Frequency Noise, Vibration and Active Control, 25(3): 195 – 207.

[4] Coley DA, (1999), An Introduction to Genetic Algorithms for Scientists and Engineers. World Scientific.

[5] Lennon WK, Passino KM (1999), Genetic adaptive identification and control. Engineering Applications of Artificial Intelligence, 12: 185-200.

[6] Kaur K, Chhabra A, Singh G (2010), Modified genetic algorithm for task scheduling in homogeneous parallel system using heuristics. *International Journal of Soft Computing*, 5(2): 42-51.

[7] Mebane WR, Sekhon JS (2011), Genetic optimization using derivatives" *Journal of Statistical Software*, 42(11): 1-26.

[8] Pham DT, Awadalla MH, Eldukhri EE (2007), Adaptive and co-operative mobile robots. Proceedings of the Institution of Mechanical Engineers, Part I: Journal of Systems and Control Engineering, 221 (3): 279-293.

[9] Kwong C, Chuah TC, Lee SW (2010), Adaptive network fuzzy inference system (ANFIS) Handoff Algorithm. *International Journal of Network and Mobile Technologies*, 1(2): 54-59.

[10] Xia L, Moore JB (1989), Recursive Identification of over parameterized systems. *IEEE Transaction on Automatic Control*, 1(34).

[11] Feng Z, Chu F, Song X (2004), Application of general regression neural network to vibration trend prediction of rotating machinery. Advances in Neural Networks - ISNN 2004, lecture Notes in Computer Science, 3174: 767-772.

[12] Ban J, Chang C(2013), The learning problem of multi-layer neural networks. *Neural Networks* 46: 116-123.

[13] Kanmani S, Uthariaraj VR, Sankaranarayanan V, Thambidurai P (2004), Object oriented software quality prediction using general regression Neural Networks. ACM SIGSOFT Software Engineering Notes, 2004, 29 (5): 1-6.

[14] Pham DT, and Castellani M (2009), The bees algorithm: modelling foraging behaviour to solve continuous optimization problems. Proceedings of the Institution of Mechanical Engineers, Part C: Journal of Mechanical Engineering Science, 223 (12), : 2919-2938.

[15] Packianather MS, Landy M, and Pham DT (2007), Enhancing the speed of the bees algorithm using pheromone-based recruitment. INDIN: 789-794.

[16] Pham DT, Ghanbarzadeh A, Koç E, Otri S , Rahim S , Zaidi M (2006), The bees algorithm – a novel tool for complex optimisation problems". Proceedings of the 2nd Virtual International Conference on Intelligent Production Machines and Systems (IPROMS 2006): 454-459.

[17] Sedighizadeh D, Masehian E, (2009), Particle swarm optimization methods. Taxonomy and Applications", International *Journal of Computer Theory and Engineering*, 1(5): 486-502.

[18] Lalwani S, Singhal S, Kumar R, and Gupta N (2013), A comprehensive survey: applications of multi-objective particle swarm optimization algorithm. *Transactions on Combinatorics*, 2 (1): 39-101.

[19] Madkour A, Hossain MA, Dahal KP, Yu Y (2007), Intelligent learning algorithms for active vibration control. IEEE Transactions On Systems, Man, and Cybernetics—Part C: Applications And Reviews, 37(5): 1022-1033.

[20] Tavakolpour AR, Mailah M, Darus IZ (2009), Active vibration control of a rectangular flexible plate structure using high gain feedback regulator. *International Review of Mechanical Engineering*, 3(5): 579-587.

[21] Tavakolpour AR (2010), Mechatronic Design of intelligent active vibration control systems for flexible structures. PhD Thesis, Faculty of Mechanical Engineering, university of technology, Malaysia.

[22] Sivanandam SN, Deepa SN (2007), Introduction to genetic algorithms. Springer-Verlag, Berlin.