ISSN 2176-5480

22nd International Congress of Mechanical Engineering (COBEM 2013)
November 3-7, 2013, Ribeirão Preto, SP, Brazil

ACTIVE VIBRATION CONTROL USING ARTIFICIAL NEURAL NETWORKS

William Camilo Ariza-Zambrano †
Alberto Luiz Serpa §
University of Campinas - UNICAMP
† ariza@fem.unicamp.br
§ serpa@fem.unicamp.br

Abstract. *This work has as main objective the study of control techniques for the problem of vibration in mechanical structures using artificial neural networks based on its inverse model. The artificial neural network controller is trained to obtain the inverse model of the plant using the Levenberg-Marquardt method from the input and output data obtained by simulation of the model when it is applied a known excitation signal. Once the training satisfies the criteria imposed, the neural network obtained is used as controller in the closed-loop cycle. The control technique was tested through the simulation of the active vibration control of a mass-spring-damper model with three degrees of freedom with two outputs (measurement output and performance output) and two inputs (control effort and disturbance input). Comparisons were made with the H_∞ control technique. The results obtained with this control scheme clearly show that the control with neural networks based on the inverse model is effective for vibration suppression in a structure.*

Keywords: *Vibrations controls, artificial neural networks, direct inverse control*

1. INTRODUCTION

The active vibration control uses the superposition of waves by generating secondary signal to attenuate the unwanted source and this result in a reduction in the level of vibration at desired location. In general this is more efficient than passive methods of vibration suppression (Darus and Tokhi, 2005).

The motivation to use artificial neural networks (ANN) in control processes is the possibility to create a model that will be able to learn from experience (Nørgaard *et al.*, 2004). It is the possibility that the experience can be interpreted as knowledge about how certain inputs affect the system, and this can be taken as advantage to solve a control problem.

This work presents a simple and intuitive control technique based on artificial neural networks to solve the problem of attenuating undesired vibration (disturbances) in a mechanical structure.

This paper is divided in four principal parts. The first section defines some concepts related to ANNs, their basic structure, training methods and finally an introduction about how it is used in a solution of a control problem. The second section goes deeper with the use of an ANN in control and introduces the *Direct Inverse Control* method based on artificial neural networks as a method for control problems. The third section describes a problem of vibration control and how it is treated according to the technique that was discussed in the preceding section. In addition, this section starts with the calculation of the plant model to control. The plant tested is a mass-spring-damper with three degrees of freedom that is exposed to a disturbance. This section ends with the results of whole control scheme. The results are compared with the H_∞ control. Finally, in the last section, there are some conclusions about this work.

2. ARTIFICIAL NEURAL NETWORKS

In general, an ANN is a system of simple processing elements, neurons, that are connected into a network by a set of (synaptic) weights (Nørgaard *et al.*, 2004). An artificial neural network can be described as mapping an input space to an output space (Priddy and Keller, 2005).

The *neuron* is the fundamental unit in a neural network. Its mathematical model is given by:

$$y_k = \varphi(\sum_{j=1}^{m} w_{kj} x_j + b_k) \tag{1}$$

The synaptic weights w_{kj} are multiplied by the inputs of the neuron x_j. There is an additional pseudo-input to the neuron, the *bias* b_k, that allows the activation function $\varphi(\bullet)$ take a value even when all inputs are zero (Haykin, 2001). The index k refers to the number of the current neuron that is being processed. The most commonly used activation functions are: the *linear* function, the *hyperbolic tangent* function, the *sigmoid* function and the *step* function.

Neurons can be combined into a network in numerous ways. One of the most common artificial neural network architecture is the *multilayer perceptron (MLP) network* (Nørgaard *et al.*, 2004). The basic MLP-network is constructed by ordering the neurons in layers. This type of networks is often referred to as *feed-forward network*, as illustrated in Figure 1.

ISSN 2176-5480

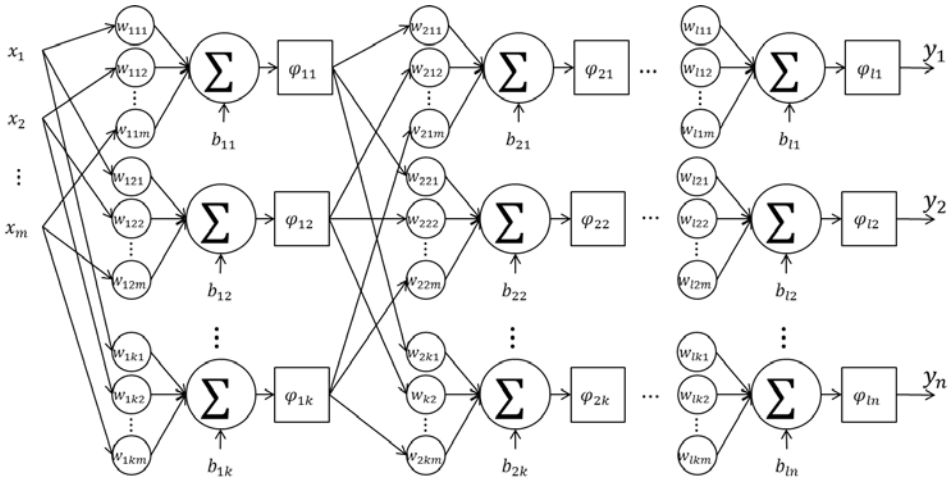


Figure 1: Multilayer perceptron network

Considering $\mathbf{X} = \{x_1, x_2, \ldots, x_m\}$ the vector with the inputs to the network and $\mathbf{Y} = \{y_1, y_2, \ldots, y_n\}$ the output network vector, the index l as the number of layers in the network, k is the number of neurons in a layer and m as the amount of inputs for each neuron, the network's output vector can be obtained with the Algorithm 1.

Algorithm 1: Outputs of MLP-network calculation

X: Network input vector
Y: Network output vector

l : Number of layers in the network
k : Vector with the numbers of neurons per layer

$x \longleftarrow \mathbf{X}$

Loop through every layer
for $i \leftarrow 1$ **to** l **do**
  Loop through every neuron in a layer
  for $h \leftarrow 1$ **to** k_i **do**
    $m \longleftarrow$ length of x
    Neuron output (See Equation (1)).
    $y_{ih} = \varphi_{ih}(\sum_{j=1}^{m} w_{ihj} x_j + b_{ih})$
  $x \longleftarrow y_{ih}$
Final network output ***Y***
$\mathbf{Y} \longleftarrow y_{ih}$

2.1 Training phase

The problem of determining the synaptic weights $\boldsymbol{W}$ (elements w_{ihj}) of every neuron connection in a artificial neural network from a input-output data-set previously obtained by and experiment or a theoretical simulation (See Figure 2) is called *training* or *learning* (Nørgaard *et al.*, 2004), and it is basically an optimization problem. The weights are obtained in such a way that the network, according to some performance measurement or cost function, models the most accurately relationship between the inputs and the outputs.

The training process for identification systems can be seen in Figure 2. $\boldsymbol{U}$ is the input signal to excite the plant obtaining the output of the process $\boldsymbol{Y}$. The $\hat{\boldsymbol{Y}}$ is the current estimated output of the network, it is compared with the system output $\boldsymbol{Y}$ through a performance criterion to feed-back the ANN in order to re-adjust the synaptic weights.

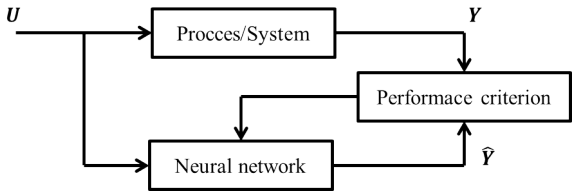


Figure 2: Neural network training scheme

ISSN 2176-5480

2.1.1 Input signal

In identification of systems with ANN there many types of input signal. The selection of a good excitation signal is an important step in the training phase. This topic is widely discussed in Pintelon and Schoukens (2005), and some relevant characteristics to choose an input signal in identification systems, like the quality of signal or broadband excitations, are shown there. Some of the excitation signals that are introduced in Pintelon and Schoukens (2005) are: the *swept sine* (also known as *periodic chirp*), the *Schroeder multi-sine*, the *pseudo-random binary sequence* (PRBS), *random noise*, *random burst*, and *pulse impact testing*.

2.1.2 Optimization problem

The training process, viewed as an optimization problem, aims to search the minimum of a criterion or cost function. The most commonly used measure of performance for this type of problems is the *mean square error* (MSE), that is calculated as:

$$\varepsilon(\boldsymbol{W}) = \frac{1}{2N}\sum_{i=1}^{N}[Y_i - \hat{Y}_i(\boldsymbol{W})]^2 \tag{2}$$

where $\boldsymbol{Y}$ is the output data-set of the system excited by an input signal $\boldsymbol{U}$ of length N, and $\hat{\boldsymbol{Y}}$ is the output data-set of the network in function of current weights matrix, $\boldsymbol{W}$, for the same input data-set $\boldsymbol{U}$.

The values of the weights matrix $\boldsymbol{W}^*$, that best fits the map from an input-data-set $\boldsymbol{U}$ to an output-data-set $\boldsymbol{Y}$, can be obtained solving the optimization problem:

$$\boldsymbol{W}^* = \min_{\boldsymbol{W}} \ \varepsilon(\boldsymbol{W}) \tag{3}$$

In ANN the most typical first-order optimization method is the *back-propagation algorithm*, that is a particular implementation of the *gradient* method.

The second-order methods uses the Hessian matrix, or an approximation of it, for example *Newton* method, *Quasi-Newton* method, *Gauss-Newton* method, *Pseudo-Newton* method and *Levenberg-Marquardt* method.

Other classes can be founded, e.g., the recursive methods, used in ANN field when it is necessary to train the network online and in general these demands more computational resources (Nørgaard *et al.*, 2004).

2.2 ANN in control systems

There are two approaches in the use of ANN in control systems: the *direct design methods* and *indirect design methods*. The first one, the *direct design*, refers to the direct implementation of the controller, and this means that the neural network must be trained as a controller according to some kind of relevant criterion. On the other hand, in the *indirect methods* the design is based on a neural network model of the system to be controlled (Nørgaard *et al.*, 2004).

2.2.1 Direct control system design

The implementation of direct control systems design can be considered simple. The design and tuning are difficult considering that the controller must be retrained every time that a design parameter changes. This method includes the following design control schemes: the *direct inverse control*, *internal model control*, *feedback linearization*, *feedforward with inverse models* and *optimal control* (Nørgaard *et al.*, 2004).

2.2.2 Indirect control system design

The idea of *indirect method* is using a neural network to model the system to be controlled. This model is then employed in a more conventional controller design. The model is typically trained in advance, but the controller is trained on-line. In this kind of design the following methods are covered: the *approximate pole placement*, *minimum variance*, *predictive control* and *non-linear predictive control* (Nørgaard *et al.*, 2004).

3. DIRECT INVERSE CONTROL

One of the most common methods to use an ANN in a control problem is *the Direct Inverse Control* (DIC). The target is to identify the inverse model of a discrete-time plant to control and use it as controller in a closed-loop system. For the

ISSN 2176-5480

identification of the inverse model, in contrast to the scheme shown in the Section 2.1 for identification systems, the ANN must estimate the control effort $\hat{u}$ applied to the plant and it has the plant output $\boldsymbol{Y}$ as input, as shown in the Figure 3.

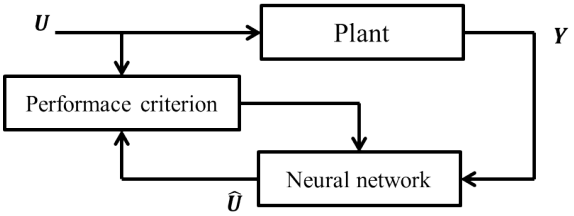


Figure 3: Inverse model identification scheme

This control scheme has the following advantages: it is intuitively simple, it is simple to implement and with specialized training the controller can be optimized for a specific reference trajectory. As disadvantages, it does not work for systems with an unstable inverse and it has problems with the inverse models that has small damping (zeros near to the unit circle). Additionally, it lacks of tuning options like response time, overshoot percentage, and generally presents a high sensitivity to disturbance and noise (Nørgaard *et al.*, 2004).

3.1 Controller design

The procedure to obtain a neuro-controller based on the *direct inverse method* scheme is divided in six principal steps. The first step is to choose the structure of the ANN that will act as controller. The second step is to generate the input-output data-sets through simulating the time-discrete plant excited with certain input signal. The data-set obtained will be used in the training phase and later in the validation phase. Then, the obtained ANN will be placed as controller to close the control loop. Finally simulations of the whole control scheme can be done.

3.1.1 Neuro-controller structure

In order that the ANN model can represent the dynamic of a discrete inverse model it is common to include a set of past outputs and inputs of the plant as network inputs (Nørgaard *et al.*, 2004). This means that the ANN will have as inputs the n past plant system outputs $\{y(k-1), y(k-2), \ldots, y(k-n)\}$ and the m past system inputs $\{u(k-1), u(k-2), \ldots, u(k-m)\}$, and as output the estimated control effort $\hat{u}(k)$, as shown in the Figure 4.

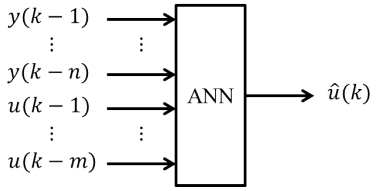


Figure 4: Neuro-controller model structure

3.1.2 Getting the training data

Some input-output data from the process to be controlled should be collected. The plant input data-set $\boldsymbol{U}$ is a vector of the *input signal* applied to the plant and it must excites the system in a operating range of interest.

The input network vector $\boldsymbol{X_{net}}$ at the instant k is formed by the actual and past values of the system output $\boldsymbol{Y}$ and the actual and the past values of the system input $\boldsymbol{U}$ (Equation (4a)). The output network vector $\boldsymbol{Y_{net}}$ at the instant k is the actual system input value $\boldsymbol{U}_k$ (Equation (4b)).

$$\boldsymbol{X_{net}}(k) = \{y(k-1), y(k-2), \ldots, y(k-n), u(k-1), u(k-2), \ldots, u(k-m)\} \tag{4a}$$

$$\boldsymbol{Y_{net}}(k) = \{u(k)\} \tag{4b}$$

With the input-output network data-sets defined the next step is the controller training phase.

3.1.3 ANN training

For the training just a fraction of the input-output network data-sets are used, generally 50% of the data-sets. The remaining fraction will be used in the validation phase. As mentioned in Section 2.1 the network learning phase is

ISSN 2176-5480

treated as a minimization problem. The output of the network $\hat{\boldsymbol{U}}$ depends on the synaptic weights matrix $\boldsymbol{W}$, that can be randomly initialized. Thus, applying to the network the input data-set $\boldsymbol{X_{net}}$, the estimated output data-set $\hat{\boldsymbol{U}}$ will be obtained. The optimal values of the synaptic weights matrix $\boldsymbol{W}^*$ are obtained by:

$$\boldsymbol{W}^* = \min_{\boldsymbol{W}} \frac{1}{2N} \sum_{i=1}^{N} [\boldsymbol{Y_{net}}_i - \hat{\boldsymbol{U}}_i(\boldsymbol{W})]^2 \tag{5}$$

The network can be trained using any training methods, e.g., *Error Back-Propagation*, *Newton algorithm*, *Gauss-Newton algorithm*, *Levenberg-Marquardt algorithm*, etc. This strategy is referred to as *general training*. There are some other procedures such the *specialized training*, which is an on-line training and has as objective the minimization of the mean square error between the signal of reference and the output of the system.

3.1.4 ANN validation

In the validation stage, the estimated model is evaluated to verify if it represents the system accordingly. The validation phase uses the data-set that was not used during training. This data-set is commonly known as *test* or *validation set*. The average error estimation can be useful to compare how good the obtained model is, but this quantity cannot directly be used for deciding if a particular model should be accepted or not. Another useful way to evaluate the ANN model is the visualization of predictions (Nørgaard *et al.*, 2004).

The average error can be estimated by evaluating the mean square error (Equation (2)) between the outputs $\boldsymbol{Y_{net}}$ of the *test set* and the network predictions $\hat{\boldsymbol{U}}$ obtained by applying the $\boldsymbol{X_{net}}$ of the *test set*.

The ANN validation based on the visualization of the predictions consists in the simply inspection of the plots comparing the outputs of *test set* $\boldsymbol{Y_{net}}$ with the predicted network outputs $\hat{\boldsymbol{U}}$. In this way, under-fitting or the over-fitting of the data can be detected (Nørgaard *et al.*, 2004).

In case of the network model is not acceptable is necessary to go back in the procedure and to train the ANN again.

3.1.5 Closed-loop system

Once the ANN inverse model is accepted the next step is to close the loop of control. The obtained network is used as controller by substituting the network input $u(k-1)$, see Figure 4, by the desired reference signal $r(k+1)$. If the network represents the exact inverse, the control input $\hat{u}(k)$ produced will drive the system output $y(k+1)$ to $r(k+1)$. This is illustrated in Figure 5.

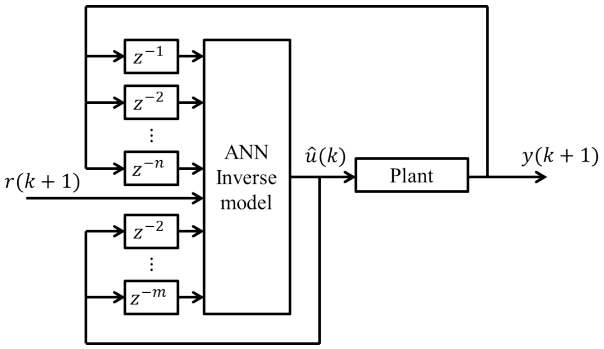


Figure 5: Direct inverse control

With the scheme shown in Figure 5 is possible to run different tests to evaluate the whole closed-loop control scheme. Depending on the resulted performance of the control scheme it will be necessary to go back to previous sections and to redefine the parameters of every stage in the controller design, until the controller shows an acceptable performance.

3.1.6 Closed-loop system responses

It is possible to run some simulations that enhance the notion about the behaviour of the closed-loop system. The analysis of the temporal response and the frequency response are extendedly used.

For the temporal response of the system it is necessary to define the external input signals of the system, such as the reference signal and disturbances if exists. Once the signals are defined, an iterative simulation of the system must be done. In every iteration k of the simulation loop, the control effort must be calculated using the trained network. This process is described in Algorithm 2:

ISSN 2176-5480

Algorithm 2: Closed-loop simulation

System model
f : System state function
h : System observation function

Simulation parameters
T_{sim} : Simulation time
T_s : Sampling period (Same as T_s model)

External inputs
$\boldsymbol{R}$ **:** Reference signals
$\boldsymbol{V}$ **:** Disturbance signals

Initial states
x_0 : Initial system state

for $k \leftarrow 0$ **to** $(T_{sim}/T_s) - 1$ **do**

$\boldsymbol{X_{net}} = \{y_{k-1}, y_{k-2}, \ldots, y_{k-n}, u_{k-1}, u_{k-2}, \ldots, u_{k-m}\}$ | In case of $k - n < 0$ then $y_{k-n} = 0$; In case of $k - m < 0$ then $u_{k-m} = 0$

$\hat{u}_k = net(\boldsymbol{W}^*, \boldsymbol{X_{net}})$ (See Algorithm 1.)

$x_{k+1} = f(x_k, \hat{u}_k, v_k)$

$y_k = h(x_{k+1}, \hat{u}_k, v_k)$

Results
$\boldsymbol{Y} \leftarrow y_{1:k}$

For the frequency system response, a data-based method must be used since we have not a single model that describes the behaviour of the closed-loop system. Estimating the frequency response of a system generally involves exciting the system with an input signal and comparing the generated data through a process such as the Fourier transformations (Nichols and Dennis, 1971).

A technique for estimating the frequency response of the system based on the spectral analysis of the input-output data-sets can be used. Those data-sets can be generated in same way as in the preceding step, Section 3.1.6. It must be considered that the input signal must cover the frequency range of interest. Generally this can be made by applying an impulse to the system (impulsive response), using a frequency sweep signal (chirp signal) or any other signal with a wide frequency spectrum.

4. DIRECT INVERSE CONTROL METHOD TEST

In order to assess the control scheme extended in the previous section, a vibrations control problem is proposed. This section starts with the deduction of the plant and control model. Thereafter, the design of the control block using the DIC method is extended and finally a presentation of results is made.

4.1 System model description

Based on the example of control of a simple mechanical system founded in Gawronski (2004), the following system of three degrees of freedom (3 DOF), Figure 6, is considered.

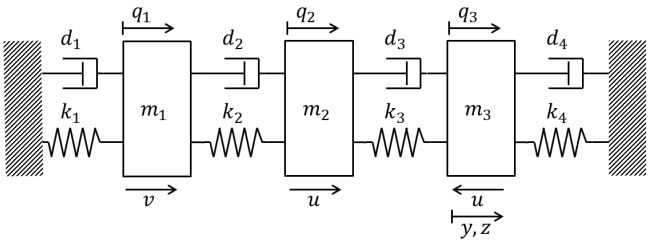


Figure 6: Plant scheme

The hypothetical numerical values of the masses are: $m_1 = 3$, $m_2 = 1$, $m_3 = 2$. The values for the stiffness are $k_1 = 30$, and $k_2 = k_3 = k_4 = 6$. The stiffness matrix $\boldsymbol{K}$ and the mass matrix $\boldsymbol{M}$ are:

ISSN 2176-5480

$$\boldsymbol{K} = \begin{bmatrix} k_1 + k_2 & -k_2 & 0 \\ -k2 & k_2 + k_3 & -k_3 \\ 0 & -k_3 & k_3 + k_4 \end{bmatrix} \quad \text{and} \quad \boldsymbol{M} = \begin{bmatrix} m_1 & 0 & 0 \\ 0 & m_2 & 0 \\ 0 & 0 & m_3 \end{bmatrix}$$

For this problem, the concept of proportional damping is used with $\alpha = 0.004$ and $\beta = 0.001$, i.e., the damping matrix is:

$$\boldsymbol{D} = \alpha \boldsymbol{K} + \beta \boldsymbol{M}$$

Considering $\boldsymbol{q} = [q_1 \ q_2 \ q_3]'$ the displacements of the masses m_1, m_2 and m_3 respectively, and $\boldsymbol{f(t)} = [f_1(t) \ f_2(t) \ f_3(t)]'$ the actuating forces in the masses, the differential equation of the system can be written as:

$$\boldsymbol{M\ddot{q} + D\dot{q} + Kq = f(t)} \qquad \text{or} \qquad \boldsymbol{\ddot{q} = -M^{-1}D\dot{q} - M^{-1}Kq + M^{-1}f(t)} \tag{6}$$

4.2 Control model description

This section starts with description of the control scheme for a multivariable system in continuous time, Figure 7. Then the obtained model is discretized with the *zero-order hold* (zoh) method in order to use it in the design of the control module.

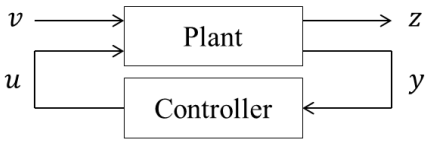


Figure 7: Multi-variable control scheme

The control objective is to decrease the disturbance (v) effect on the system, located as a force actuating in the mass m_1, using an actuator placed between the masses m_2 and m_3. The performance measurement z is the displacement of mass m_3 that is also used as measurement for the controller (see Figure 6).

For this system, the state space vector can be:

$$\boldsymbol{x} = \left\{ \begin{array}{c} \boldsymbol{q} \\ \boldsymbol{\dot{q}} \end{array} \right\} = [q_1 \ q_2 \ q_3 \ \dot{q}_1 \ \dot{q}_2 \ \dot{q}_3]' \tag{7}$$

So, the state-space model for the Figure 7 is:

$$\begin{array}{rcccccc} \boldsymbol{\dot{x}} = & \boldsymbol{Ax} & + & \boldsymbol{B_1 v} & + & \boldsymbol{B_2 u} \\ \boldsymbol{z} = & \boldsymbol{C_1 x} & + & \boldsymbol{D_{11} v} & + & \boldsymbol{D_{12} u} \\ \boldsymbol{y} = & \boldsymbol{C_2 x} & + & \boldsymbol{D_{21} v} & + & \boldsymbol{D_{22} u} \end{array} \tag{8}$$

Then, taking the Equation (6) it is possible to write:

$$\boldsymbol{\dot{x}} = \left\{ \begin{array}{c} \boldsymbol{\dot{q}} \\ \boldsymbol{\ddot{q}} \end{array} \right\} = \underbrace{\begin{bmatrix} \boldsymbol{0} & \boldsymbol{I} \\ \boldsymbol{-M^{-1}K} & \boldsymbol{-M^{-1}D} \end{bmatrix}}_{\boldsymbol{A}} \left\{ \begin{array}{c} \boldsymbol{q} \\ \boldsymbol{\dot{q}} \end{array} \right\} + \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{M^{-1}f(t)} \end{bmatrix}$$

The vector $\boldsymbol{B_{0v}}$ is the vector of the actuating disturbing forces on the masses m_1, m_2 and m_3. As the disturbance $\boldsymbol{v}$ is one force actuating over m_1 then, we have that:

$$\boldsymbol{B_{0v}} = \left\{ \begin{array}{c} 1 \\ 0 \\ 0 \end{array} \right\} \quad \text{and} \quad \boldsymbol{B_1} = \begin{bmatrix} \boldsymbol{0} \\ \boldsymbol{-M^{-1}B_{0v}} \end{bmatrix}$$

ISSN 2176-5480

In the same way, the vector $\boldsymbol{B_{0u}}$ is defined. The controlling effort $\boldsymbol{u}$ is one actuator placed between the masses m_2 and m_3, that means they are opposing forces, so:

$$\boldsymbol{B_{0u}} = \left\{ \begin{array}{c} 0 \\ 1 \\ -1 \end{array} \right\} \quad \text{and} \quad \boldsymbol{B_2} = \begin{bmatrix} \mathbf{0} \\ -\boldsymbol{M}^{-1}\boldsymbol{B_{0u}} \end{bmatrix}$$

The performance measurement $\boldsymbol{z}$ is taken from the m_3 position, so that:

$$\boldsymbol{C}_1 = [0\ 0\ 1\ 0\ 0\ 0] \quad , \quad \boldsymbol{D}_{11} = 0 \quad \text{and} \quad \boldsymbol{D}_{12} = 0$$

In the same fashion, the measurement $\boldsymbol{y}$ used by the controller, is;

$$\boldsymbol{C}_2 = [0\ 0\ 1\ 0\ 0\ 0] \quad , \quad \boldsymbol{D}_{21} = 0 \quad \text{and} \quad \boldsymbol{D}_{22} = 0$$

The next phase is to obtain a discrete-time model to be used in the neuro-controller design phase. For this is used a discretization with the *zero-order hold* (zoh) method that assumes that the control inputs are constant over the sampling period T_s. This work uses the ©MATLAB implemetation of this method (See *c2d* function documentation) with a $T_s = 0.2.$ seconds.

4.3 Neuro-controller desing

For the development of the ANN that will act as controller the procedure defined in Section 3 is followed. The first step is to choose the neuro-controller structure. Afterwards, the input signal to excite the plant must be selected and constructed to generate the data-sets for subsequent training and validation of the model. Finally, a simulation of the closed-loop system must be done.

4.3.1 Controller structure

To build the ANN, the first consideration is to choose the number of past samples of the output signal $\boldsymbol{y}$ and the number of the past values of the control signal $\boldsymbol{u}$ that will be used as network inputs. For this case $m = 10$ and $n = 10$ (see Figure 4).

4.3.2 Training and validation data-set

The first step to get the training and validation data-set is to define the input signal to excite the plant. In Nørgaard *et al.* (2004) an extension of the *random noise* signal called *level change at random instances* signal is introduced, and this is suitable for use in this work.

The control model defined in Section 4.2 has two inputs. For collecting the input-output data-set the disturbance input $\boldsymbol{v}$ will be considered as zero. To construct the signal, it is defined the *N-samples-constant* signal as:

$$u(k) = e\left(\text{int}\left[\frac{k-1}{N}\right] + 1\right) \tag{9}$$

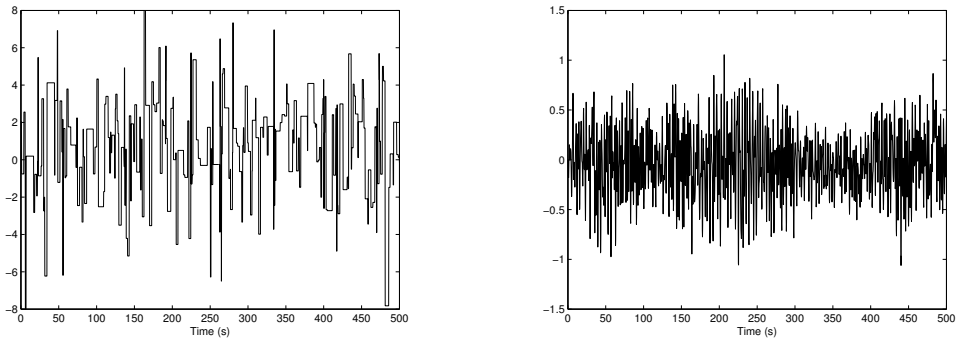
with $e(\bullet)$ a white noise signal with variance σ_e^2. The function int denotes the integer part and N is the number of samples. The *level change at random instances* signal introduces the additional random variable α for deciding when to change the level, so the input signal can be obtained by:

$$u(k) = \left\{ \begin{array}{ll} u(k-1) & \text{with probability} \quad \alpha \\ e(k) & \text{with probability} \quad 1-\alpha \end{array} \right. \tag{10}$$

The Figure 8a shows the *level change at random instances* signal used to simulate the system with $\alpha = 0.9$ and $\sigma_e^2 = 8$. The signal is created for a total time of 500 seconds taking into account the sampling period of 0.2 seconds. For the required data-sets, the measurement output $\boldsymbol{y}$ is obtained with the control signal $\boldsymbol{u}$ defined as in Figure 8a and $\boldsymbol{v} = 0$. The output system $\boldsymbol{y}$ shown in Figure (8b).

It is possible to get the data-sets $\boldsymbol{X_{net}}$ and $\boldsymbol{Y_{net}}$ respectively following the Equation (4a) and Equation (4b).

ISSN 2176-5480



(a) *Level change at random instances* signal (b) System output y

Figure 8: Training and validation data-set

4.3.3 Training

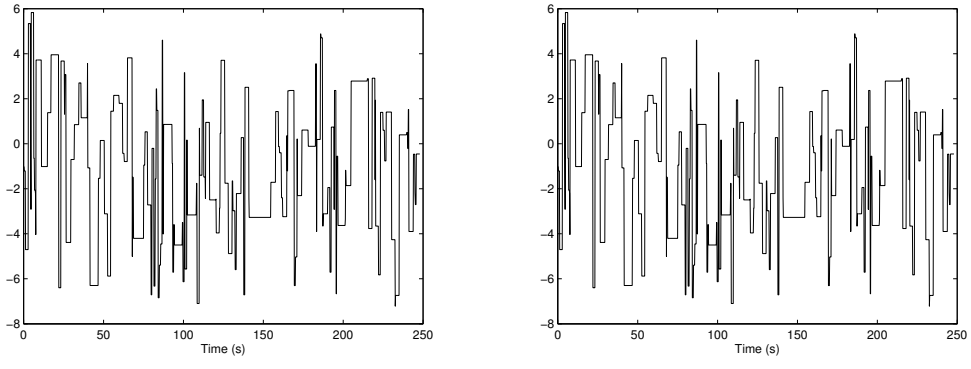
The *Levenberg-Marquardt* method is the algorithm used in this work according to the Nørgaard (2000) toolkit. It provides a numerical solution to the minimization problem. In ANN, this algorithm is suitable for training small and medium-sized problems (Wilamowski and Irwin, 2011).

Half of the defined data-sets are taken for training. In 500 iterations the network performance (MSE, Equation (2)) converged to 2×10^{-10}.

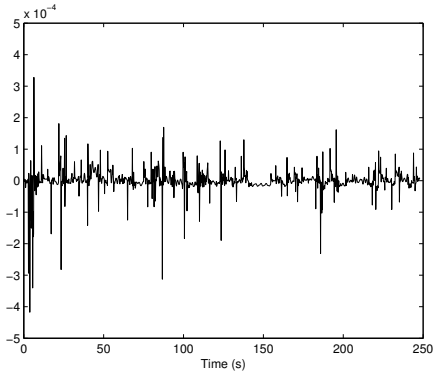
4.3.4 Validation

The remaining half data-set are used for validation. With this data-set the mean square error is around to 7×10^{-10}.

As mentioned in Section 3.1.4, the validation based on the visualization of the predictions is the comparison of the plots of the predicted outputs and the data-set outputs. The network predictions, Figure (9b), is quite similar to the actual control signal, Figure (9a). The prediction error values $u - \hat{u}$, Figure (9c), are low as expected.



(a) Actual control signal (b) ANN prediction control signal



(c) Prediction error ($u - \hat{u}$)

Figure 9: Validation of the model

ISSN 2176-5480

The validation phase shows that the ANN inverse model obtained is an acceptable predictor of the control effort applied to the plant. To evaluate the ANN inverse model as controller it will be necessary to run the simulation of the whole closed-loop system for subsequent results analysis.

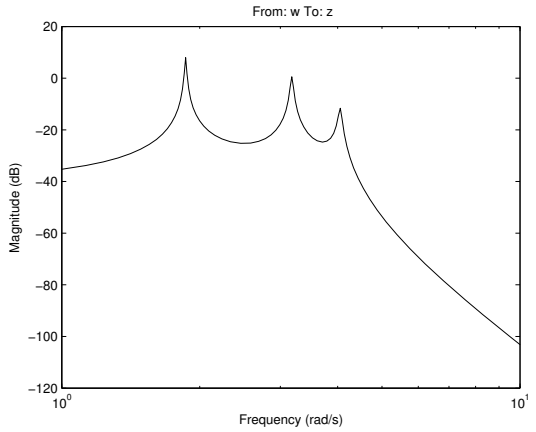
4.4 H_∞

In order to make a comparison of how effective is the proposed control scheme, the problem at hand is solved by the control H_∞ technique. Weighting filters were not considered for the performance outputs of the system $\boldsymbol{z}$ (See Figure 7). The H_∞ solution was founded with help of the ©MATLAB robust control toolbox. The function *dhinf* was used.

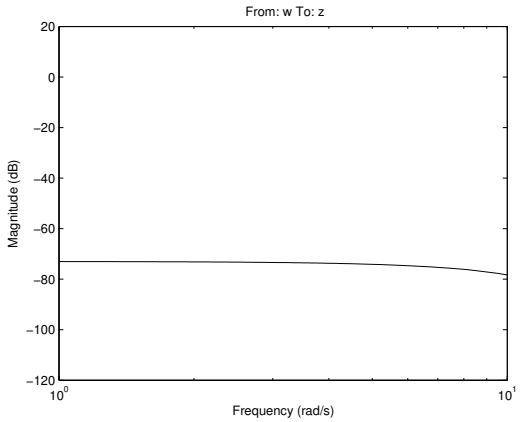
4.5 Results

The benchmark to compare both, direct inverse model and H_∞ control solutions, includes a disturbance signal of type *chirp signal*, with amplitude between $[-1, 1]$ with frequencies from $f_{\text{min}} = 1\ [rad/s]$ to $f_{\text{max}} = 7\ [rad/s]$, applied to the plant for 400 seconds. The temporal and frequency responses were obtained.

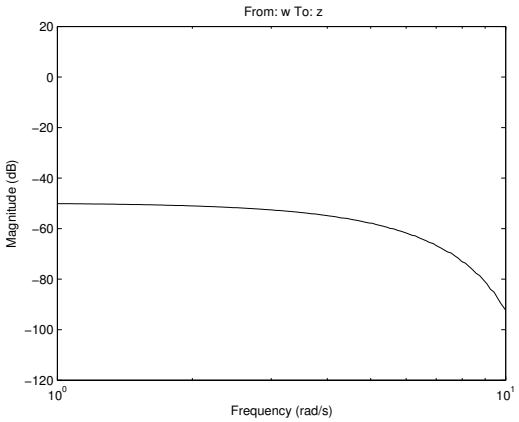
In the Figure 10a the plant frequency response is shown. It is clearly seen the three peaks corresponding to the three modes of vibration of the structure to be controlled. In the case of the solution by H_∞ scheme (Figure 10b), it is shown that the closed-loop system starts attenuating the signal in around $-70\ [dB]$ to $-80\ [dB]$ close to the $10\ [rad/s]$. The *direct inverse control* (DIC) scheme (Figure 10c) was effective, starting with an attenuation of $-50\ [dB]$ at $1\ [rad/s]$ and it comes close to the $-100\ [dB]$ of attenuation near to the $10\ [rad/s]$.



(a) Open-loop system / Plant frequency response



(b) Closed-loop with H_∞ control



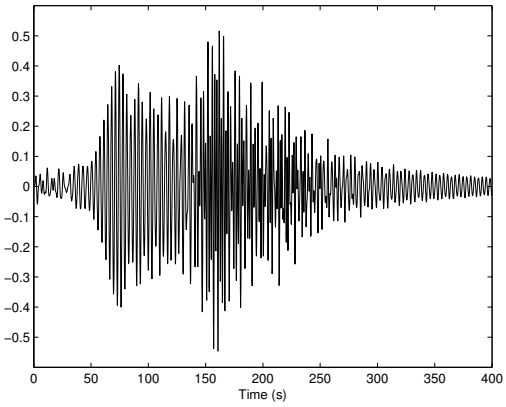
(c) Closed-loop with *DIC* control

Figure 10: System frequency response

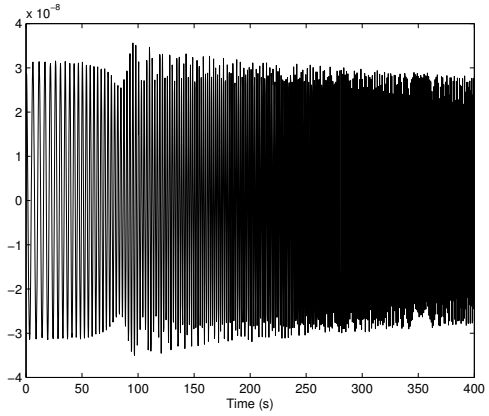
The open-loop system temporal response (Figure 11a) shows the expected peaks shown in the open-loop system frequency response (Figure 10a). Both, the H_∞ controlled system (Figure 11b) and the neuro-controlled system temporal responses (Figure 11c) present a significant attenuation of the disturbance signal response.

In this case, it is expected to have an excellent result for the H_∞ control since the model is precisely determined. It is possible to verify that the DIC presented a comparable result to the H_∞ control.

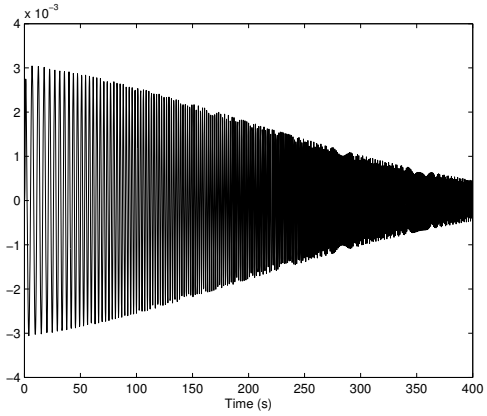
ISSN 2176-5480



(a) Open-loop system / Plant response



(b) Closed-loop with H_∞ control



(c) Closed-loop with *DIC* control

Figure 11: System temporal response

ISSN 2176-5480

5. CONCLUSIONS

A procedure to obtain a neuro-controller based on the inverse system model was shown in this work. The results of the direct inverse control scheme based on artificial neural network shows that it can be used as an effective method in the active vibration control problem, given a satisfactory response of the system in the operating condition treated. In addition, the scheme presented is a model-data-based method, therefore, it can be very useful in control cases that there is not *a priori* information about the system. The procedure of neural control indicated in this paper can be considered the simplest scheme in neural control. Future work may address control techniques as optimal control based in ANN or predictive control as well.

6. ACKNOWLEDGEMENTS

Special thanks to the *Programa de Apoio ao Ensino e à Pesquisa Científica e Tecnológica em Engenharias* (Pró-Engenharias/CAPES) for the financial support.

7. REFERENCES

Darus, I.M. and Tokhi, M., 2005. "Soft computing-based active vibration control of a flexible structure". *Engineering Applications of Artificial Intelligence*, Vol. 18, pp. 93 – 114.

Gawronski, W.K., 2004. *Advanced Structural Dynamics and Active Control of Structures*. Springer.

Haykin, S., 2001. *Redes neurais: Princípios e prática*. Bookman, 2nd edition.

Nichols, S. and Dennis, L., 1971. "Estimating frequency-response function using periodic signals and the f.f.t." *Electronics Letters*, Vol. 7, No. 22, pp. 662–663. ISSN 0013-5194. doi:10.1049/el:19710452.

Nørgaard, M., Ravn, O., Poulsen, N. and Hansen, L., 2004. *Neural Networks for Modelling and Control of Dynamic Systems: A Practitioner's Handbook*. Springer.

Nørgaard, M., 2000. "Neural network based control system design toolkit, ver 2". Technical report, Department of Automation, Technical University of Denmark.

Pintelon, R. and Schoukens, J., 2005. *System Identification: A Frequency Domain Approach*. Chapter 4. Design of Excitation Signals. John Wiley & Sons, Inc.

Priddy, K.L. and Keller, P.E., 2005. *Artificial neural networks: An introduction*. SPIE - The International Society for Optical Engineering.

Wilamowski, B.M. and Irwin, J.D., 2011. *Industrial Electronics Handbook: Intelligent Systems*, Vol. 5 of *Chapter 12 - Hao Yu and B. M. Wilamowski, "Levenberg−Marquardt Training"*. CRC Press, 2nd edition.

8. RESPONSIBILITY NOTICE

The authors are the only responsible for the printed material included in this paper.