Thai Journal of Mathematics
Volume 21 Number 1 (2023)
Pages 183–199

http://thaijmath.in.cmu.ac.th

ISSN 1686-0209

Artificial Neural Network Model for Analysis of Linear Dynamic Systems Subject to Non-stationary Excitations

Pawarid Posayanant[1], Patiphan Chantarawichit [2], Damang Dy[3] and Yos Sompornjaroensuk[1,*]

[1] *Department of Civil Engineering, School of Engineering and Industrial Technology, Mahanakorn University of Technology, Thailand*
e-mail : 6219310001@mutacth.com (P. Posayanant); syossyos@mutacth.com (Y. Sompornjaroensuk)

[2] *Department of Civil Engineering, Faculty of Engineering, Rajamangala University of Technology Krungthep, Thailand*
e-mail : patiphan.c@mail.rmutk.ac.th (P. Chantarawichit)

[3] *General Department of Immigration, Ministry of Interior,National Road Number 1, S/K Nirouch, Chbar Ampov, Phnom Penh 12355, Cambodia*
e-mail : dydamang@gmail.com (D. Dy)

Abstract Computation of stochastic responses is a key step in reliability and probabilistic analysis of dynamic systems. Monte Carlo Simulation (MCS) is generally employed for accurate analysis. Artificial Neural Network (ANN) has a capability of mapping input to output. Due to the requirement of large sample sizes in the reliability and probabilistic analysis, ANN has been successfully applied as a surrogate model in many applications except non-stationary excitations. This paper proposes for the first time to apply ANN as the surrogate model for the non-stationary excitation. Specifically, multi-layer feed-forward ANN is employed for the purpose. The applicability of the proposed methodology is illustrated through a probabilistic analysis of a 3-DOF linear system subjecting to non-stationary ground excitation. The numerical results shown the potential of the proposed methodology.

MSC: 68T07

Keywords: artificial neural network; linear dynamic analysis; non-stationary excitation; reliability and probabilistic analysis; surrogate model

Submission date: 18.02.2022 / Acceptance date: 01.06.2022

1. Introduction

Computation of responses at particular time instances is generally encountered in reliability and probabilistic analysis of linear dynamic systems subject to non-stationary excitations. Monte Carlo Simulation is utilized in such analysis in order to obtain accurate results. Accordingly, a number of non-stationary excitation realizations is generated

*Corresponding author.

in accordance with probabilistic descriptions, input to the system models, and perform dynamic analysis. When rare events are of interest, e.g. the probabilities of event occurrences at the order of 10^{-6} or lower, the computation requires an extremely large number of non-stationary excitation realizations, i.e. at the order of 10^{6} or higher for the exemplified cases above. The extremely large number of non-stationary excitation realizations is then input to the system models. Dynamic analysis is then performed to obtain system response. When a large number of time steps are used in the response computation, the computational effort is considerably high.

Computational intelligence has been applied to various domains (Harnpornchai et al. [1]; Paokanta et al. [2]; Paokanta and Harnpornchai [3]; Paokanta et al. [4]; Paokanta et al. [5] Harnpornchai and Wonggattaleekam [6],[7]). One of well-known computational intelligence models is Artificial Neural Network (ANN). ANN has a capability of mapping input to output. This is due to the ability of universal function approximation [8]. In this paper, ANN is employed as a mapping function from non-stationary excitations to their corresponding responses at particular time instances. More specifically, a feed-forward ANN is used for the purpose. Accordingly, desired system responses are obtained by forward computation with non-stationary excitations as input, which is different from a recursive computation in traditional time domain analysis.

Therefore, the computational effort is apparently reduced. In context of reliability and probabilistic analysis, the proposed ANN can be considered as a surrogate model of performance function containing several random variables each of which is an excitation magnitude at specific time instance. Such sequential random variables compose a non-stationary stochastic excitation according to probability and stochastic process theory. The application of ANN as a surrogate model of performance function can be found in several researchers [9–13]. Most recent applications of ANN surrogate models include the performance assessment of a vertical structure subjected to non-stationary, tornadic wind loads [14], the simulation of wave propagation [15], the analysis of mooring lines and risers [16], the prediction of pavement response to various tire configurations [17], estimation of axial load-carrying capacity of concrete-filled steel tubes [18], self-compacting concrete strength prediction [19], etc. However, it is the first time that this paper employs ANN for mapping non-stationary excitations to stochastic dynamic responses.

The structure of paper is as follows. Following this introduction, the description of ANN is presented. The non-stationary excitation model of ground acceleration and the time-domain analysis of linear dynamic system are respectively described.

The learning of ANN for modeling the mapping function of non-stationary excitations to dynamic responses at particular time instances is next explained. The proposed idea and computational procedure is shown using numerical examples. Finally, the conclusion is made.

2. Artificial Neural Network(ANN)

ANN is made up of simple processing units each of which is called neuron. The neurons can store knowledge and apply it later. ANN emulate the brain in two aspects:

1) ANN obtains knowledge through learning processes.
2) The acquired knowledge is stored in terms of synaptic weights.

2.1 Neuron

A neuron is an information-processing unit and is the basic for the ANN operation. A neuron k is depicted as shown in Figure 1.

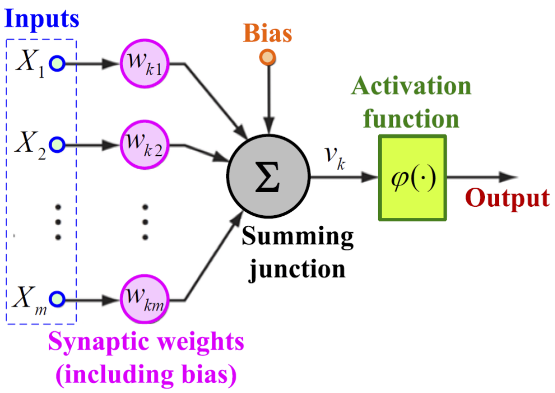

FIGURE 1. Model of a neuron.

A neuron contains three important elements:

(a) Synapses or connecting links. Each synapse is associated with a weight. The input at the synapse j connected to the neuron k is modified by the synaptic weight w_{kj}. The synaptic weight is normally a real value.

(b) Adder for summing the modified input from applying the synaptic weight. The summation is thus in the form of a linear combination.

(c) Activation function which is used for limiting the amplitude of the neuron output. The activation function is also known as a squashing function because it limits the range of output to be within an interval. Generally, the interval is $[0, 1]$ or $[-1, 1]$.

An external bias is also included to the neural representation. The bias is denoted by b_k. The effect of bias is to modify the linear combination of all inputs, that can be in the increasing or decreasing manner.

Let the linear combination of all inputs be u_k, i.e.

$$u_k = \sum_{j=1}^{m} w_{kj} x_j \tag{2.1}$$

where x_j is the j-th input.

The output from the neuron is obtained from applying the activation function, i.e.

$$y_k = \varphi(u_k + b_k) \tag{2.2}$$

where $\varphi(\cdot)$ is the activation function.

Note that the total input to the activation function v_k is equal to

$$v_k = u_k + b_k. \tag{2.3}$$

2.2 Activation Function

Typical activation functions are:

(a) Threshold function or Heaviside function.

(b) Sigmoid function or the socalled S-shaped function. The sigmoid function is the most commonly used activation function class used in the architecture of ANN. The activation functions in the class of sigmoid function consist of:

Logistic Function

$$\varphi(v) = \frac{1}{1+\exp(-av)} \tag{2.4}$$

in which a is the slope parameter.

Hyperbolic Tangent Function

$$\varphi(v) = \tanh(v). \tag{2.5}$$

The hyperbolic tangent function can yield negative as well as positive value and thus very practical compared with the logistic function.

2.3 ANN Architecture

Typical ANN architectures are:

(a) Single-Layer Feed-Forward ANN.

(b) Multi-Layer Feed-Forward ANN.

(c) Recurrent ANN.

The Multi-Layer Feed-Forward ANN which works as a mapping function will be employed in this paper. Specifically, the Multi-Layer Feed-Forward ANN is composed of one input layer, at least one hidden layer, and one output layer. The Multi-Layer Feed-Forward ANN is graphically shown in Figure 2.

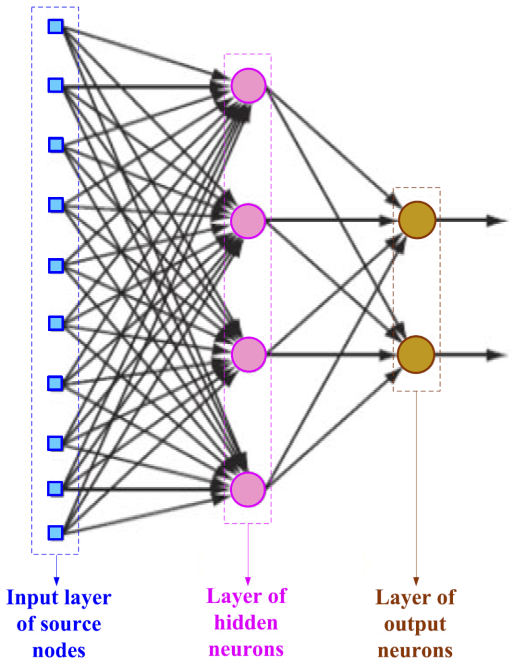

Figure 2. Multi-Layer Feed-Forward ANN.

2.4 ANN Learning

The Multi-Layer Feed-Forward ANNs are capable of representing a broad class of continuous functions including linear and non-linear forms. The determination of the synaptic weights requires higher computational effort when there are many hidden layers and neurons. An efficient learning method of weight determination is the so-called back propagation algorithm. This algorithm has been extensively used by both research and practice communities in various applications. The algorithm was invented and rediscovered before it established the recognition through the PDP group work in 1985 [20]. The algorithm has been one of the most cited and applied for ANN learning since then. Accordingly, the back-propagation algorithm will be used in this paper. The details of the algorithm can be found in the literature, e.g. [21].

The back propagation (BP) algorithm was proposed in 1986 by Rumelhart, Hinton and Williams for setting weights and hence for the training of multi-layer ANNs. This opened the way for using multi-layer ANNs, nothing that the hidden layers have no desired (hidden) outputs accessible. Once the BP algorithm of Rumelhart et al. was published, it was very close to algorithms proposed earlier by Werbos in his Ph.D. dissertation in Harvard in 1974 and then in a report by D. B. Parker at Stanford in 1982, both unpublished and thus unavailable to the community at large. It goes without saying that the availability of a rigorous method to set intermediate weights, namely to train hidden layers of ANNs gave a major boost to the further development of ANN, opening the way to overcome the single-layer shortcomings that had been pointed out by Minsky and which nearly dealt a death blow to ANNs.

The BP algorithm starts, of necessity with computing the output layer, which is the only one where desired outputs are available, but the outputs of the intermediate layers are unavailable (see Figure 3), as follows: Let ε denote the error-energy at the output layer, where:

$$\varepsilon \triangleq \frac{1}{2}\sum_k (d_k - y_k)^2 = \frac{1}{2}\sum_k e_k^2 \; k = 1, \cdots, N \tag{2.6}$$

N being the number of neurons in the output layer. Consequently, a gradient of ε is considered, where:

$$\nabla\varepsilon_k = \frac{\partial\varepsilon}{\partial w_{kj}}. \tag{2.7}$$

Using the steepest descent (gradient) one has

$$w_{kj}(m+1) = w_{kj}(m) + \Delta w_{kj}(m) \tag{2.8}$$

J denoting the jth input to the kth neuron of the output layer, where, again by the steepest descent procedure:

$$\Delta w_{kj} = -\eta\frac{\partial\varepsilon}{\partial w_{kj}}. \tag{2.9}$$

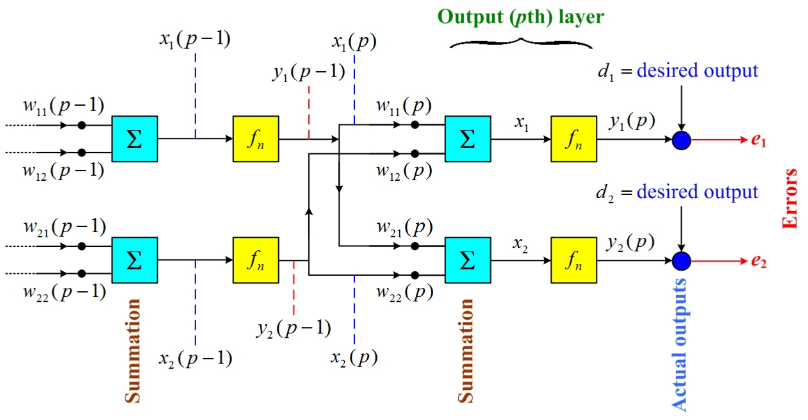

FIGURE 3. An Error at the output layer.

The minus (−) sign in Eq.(2.9) indicates a down-hill direction towards a minimum. We note from the perceptron's definition that the k's perceptron's node-output $\mathrm{z_k}$ is given by

$$z_k = \sum_j w_{kj} x_j \tag{2.10}$$

x_j being the j$^{\text{th}}$ input to that neuron, and noting that the perceptron's output y_k is:

$$y_k = F_N\left(Z_k\right) \tag{2.11}$$

F being a nonlinear function and must be continuous to allow its differentiation. We now substitute

$$\frac{\partial \varepsilon}{\partial w_{kj}} = \frac{\partial \varepsilon}{\partial z_k}\frac{\partial z_k}{\partial w_{kj}} \tag{2.12}$$

and, by Eq.(2.10)

$$\frac{\partial z_k}{\partial w_{kj}} = x_j(p) = y_j(p-1) \tag{2.13}$$

p denoting the output layer, such that Eq.(2.12) becomes:

$$\frac{\partial \varepsilon}{\partial w_{kj}} = \frac{\partial \varepsilon}{\partial z_k} x_j(p) = \frac{\partial \varepsilon}{\partial z_r} y_j(p-1). \tag{2.14}$$

Defining:

$$\Phi_k(p) = -\frac{\partial \varepsilon}{\partial z_k(p)}. \tag{2.15}$$

Then Eq.(2.14) yields:

$$\frac{\partial \varepsilon}{\partial w_{kj}} = -\Phi_k(p) x_j p = -\Phi_k y_i(p-1) \tag{2.16}$$

and, by Eqs.(2.9) and (2.16):

$$\Delta w_{kj} = \eta \phi_k(p) x_j(p) = \eta \Phi_k(p) y_j(p-1) \tag{2.17}$$

j denoting the jth input to neuron k of the output (p) layer. Furthermore, by Eq.(2.15)

$$\Phi_k = -\frac{\partial \varepsilon}{\partial z_k} = -\frac{\partial \varepsilon}{\partial y_k}\frac{\partial y_k}{\partial z_k}. \tag{2.18}$$

From Eq.(2.16)

$$\frac{\partial \varepsilon}{\partial y_k} = -\left(d_k - y_k\right) = y_k - d_k \tag{2.19}$$

whereas, for a sigmoid activation function:

$$y_k = F_N\left(z_k\right) = \frac{1}{1+\exp\left(-z_k\right)}. \tag{2.20}$$

Therefore,

$$\frac{\partial y_k}{\partial z_k} = y_k\left(1-y_k\right). \tag{2.21}$$

Consequently; by Eqs.(2.18), (2.19) and (2.20)

$$\Phi_k = y_k\left(1-y_k\right)\left(d_k - y_k\right) \tag{2.22}$$

such that, at the output layer, by Eqs. (2.9), (2.12)

$$\Delta w_{kj} = -\eta\frac{\partial \varepsilon}{\partial w_{kj}} = -\eta\frac{\partial \varepsilon}{\partial z_k}\frac{\partial z_k}{\partial w_{kj}} \tag{2.23}$$

where, by Eqs.(2.13) and (2.18)

$$\Delta w_{kj}(p) = \eta\Phi_k(p)y_j(p-1) \tag{2.24}$$

Φ_K being as in Eq.(2.22), to complete the derivation of the setting of output layer weights.

Back-propagating to the rth hidden layer, we still have, as before

$$\Delta w_{ji} = -\eta\frac{\partial \varepsilon}{\partial w_{ji}} \tag{2.25}$$

for the ith branch into the jth neuron of the rth hidden layer. Consequently, in parallelity to Eq.(2.12)

$$\Delta w_{ji} = -\eta\frac{\partial \varepsilon}{\partial z_j}\frac{\partial z_j}{\partial w_{ji}} \tag{2.26}$$

and noting Eq.(2.13) and the definition of Φ in Eq.(2.18):

$$\Delta w_{ji} = -\eta\frac{\partial \varepsilon}{\partial z_j}y_i(r-1) \tag{2.27}$$

such that, by the right hand-side relation of Eq.(2.18)

$$\Delta w_{ji} = -\eta\left[\frac{\partial \varepsilon}{\partial y_i(r)}\frac{\partial y_j}{\partial z_j}\right]y_i(r-1), \tag{2.28}$$

where $\frac{\partial \varepsilon}{\partial y_j}$ is inaccessible (as is, therefore, also $\Phi_j(r)$ above).

ε can only be affected by upstream neurons when one propagates back-wards from the output. No other information is available at that stage. Therefore

$$\frac{\partial \varepsilon}{\partial y_j(r)} = \sum_k \frac{\partial \varepsilon}{\partial z_k(r+1)}\left[\frac{\partial z_k(r+1)}{\partial y_j(r)}\right] = \sum_k \frac{\partial \varepsilon}{\partial z_k}\left[\frac{\partial}{\partial y_j(r)}\sum_m w_{km}(r+1)y_m(r)\right] \tag{2.29}$$

where the summation over k is carried out over the neurons of the next (the $r+1$) layer that connect to $y_j(r)$, while the summation over m is done over all inputs to each k' th neuron of the $(r+1)$ layer.

The definition of $\Phi, Eq.(29)$ yields

$$\frac{\partial\varepsilon}{\partial y_j(r)} = \sum_k \frac{\partial\varepsilon}{\partial z_k(r+1)} w_{kj} = -\sum_k \Phi_k(r+1)w_{kj}(r+1) \tag{2.30}$$

because only $w_{kj}(r+1)$ is connected to $y_j(r)$.

Accordingly, by Eqs.(2.18), (2.19) and (2.30):

$$\Phi_j(r) = -\frac{\partial y_i}{\partial z_j}\sum_k \Phi_k(r+1)w_{kj}(r+1) = y_j(r)\left[1-y_j(r)\right]\sum_k \Phi_k(r+1)w_{kj}(r+1) \tag{2.31}$$

and, via Eq.(2.24):

$$\Delta w_{kj}(r) = \eta\Phi_j(r)y_j(r-1)$$

to obtain $\Delta w_{ji}(r)$ as a function of ϕ and the weights of the $(r+1)$ layer, noting Eq.(2.30). Note that the partial derivatives of ε cannot be taken with respect to the hidden layer considered. It thus must be taken as the partial derivatives of ε with respect to the variables upstream in the direction of the output, which are the only ones that affect ε. This observation is the basis for the BackPropagation procedure, to facilitate overcoming the lack of accessible error data in the hidden layers.

The BP algorithm thus propagates backwards all the way to $r=1$ (the first layer), to complete its derivation. Its computation can thus be summarized as follows:

Apply the first training vector. Subsequently, compute $\Delta w_{kj}(p)$ from Eqs.(2.22) and (2.23) for the output (the p) layer and then proceed through computing $\Delta w_{ji}(r)$ from Eq.(2.31) for $r = p-1, p-\ 2,\ldots,2,1$; using Eq.(2.30) to update $\Phi_j(r)$ on the basis of $\Phi_j(r+1)$ upstream (namely back-propagating from layer $r+1$ to layer r), etc. Next, update $w(m+1)$ from $w(m)$ and $\Delta w(m)$ for the $m+1$ iteration via Eq.(2.8) for the latter training set. Repeat the whole process when applying the next training vector until you go through all L training vectors. Then repeat the whole process for $(m+2)$,$(m+3)$, $\cdots$. until adequate convergence is reached.

Initialization of $w_{ji}(0)$ is accomplished by setting each weight to a low-valued random value selected from a pool of random numbers, say in the range from -5 to $+5$.

3. Non-stationary Group Excitation Model and Simulation

In this section, the model of non-stationary excitation model and its simulation is described.

3.1 Stationary Kanai-Tajimi Model

The Power Spectral Density Function (PSDF) resulting from the filtering of strong ground motion originated at the bedrock and propagating through the soil layer is defined by [22, 23]

$$S(w) = S_0\frac{\omega_g^4 + 4\eta_g^2\omega_g^2\omega^2}{\left(\omega_g^2-\omega^2\right)^2 + 4\eta_g^2\omega_g^2\omega^2} \tag{3.1}$$

where s_0 is the intensity of the PSDF at the rock level, η_g and ω_g are the damping and the natural frequency of the soil, respectively.

3.2 Non-stationary and Evolutionary Models of PSDF

The non-stationary PSDF of the ground acceleration can be obtained from applying the modulation function to the stationary PSDF, i.e.

$$S(t,w) = \left|A\left(t,\omega^2\right)\right| S(\omega) \tag{3.2}$$

where $A(t,\omega)$ is a modulation function and $S(\omega)$ is a stationary PSDF, e.g. as given by (3.1).

When $A(t,\omega)$ is a separable function of time and frequency function, the non-stationary PSDF as given by (3.2) is reduced to the well-known uniformly modulated non-stationary random processes. This kind of processes characterizes the invariant PSDF at every time instance. Its entropy is thus constant and the analysis follows the same procedures for stationary stochastic processes.

One of the widely applied modulation function is

$$A(t,\omega) = e(t)g(t,\omega) = A_0\left[e^{-\alpha t} - e^{\beta t}\right] e^{\frac{(-r\omega t)}{\omega_m t_m}} \tag{3.3}$$

where $A_0, \alpha, \beta, r, \omega_m, t_m$ are constants. It can be shown that the entropy of the non-stationary processes using the aforementioned modulation function (3.3) is time variant. The spectral representations (3.1) and (3.2) together with the modulation function (2.8) will be used herein to construct a non-stationary stochastic process.

3.3 Simulation of Non-Stationary Ground Motion Processes

(a) Spectral Representation of Non-stationary Stochastic Processes Based on Priestley's theory of evolutionary spectral representation of non-stationary stochastic processes [24, 25], an uni-variate and one dimensional non- stationary stochastic processes, the time series representation is

$$f_0(t) = \sqrt{2}\sum_{k=0}^{\infty}\sqrt{2S_{f_0f_0}\left(t,\omega_k\right)\Delta\omega}\cos\left(\omega_k t + \phi_k\right). \tag{3.4}$$

(b) Simulation of Non-stationary Stochastic Processes

From the infinite series representation shown in (3.4), it follows that the non-stationary stochastic process $f_0(t)$ can be simulated by the following formula

$$f(t) = \sqrt{2}\sum_{k=0}^{N-1}\sqrt{2S_{f_0f_0}\left(t,\omega_n\right)\Delta\omega}\cos\left(\omega_n t + \phi_n\right) \tag{3.5}$$

where $\omega_n = n\Delta\omega,\ n = 0,1,2,\ldots,N-1$ and $\Delta\omega = \omega_u/N$.

It is assumed that

$$S_{f_0f_0}(t,0) = 0 \tag{3.6}$$

where ω_u represents an upper cut-off frequency which may be determined according to precision requirement and beyond ω_u the evolutionary power spectral density function may be assumed to be zero. As such, ω_u is a fixed value and hence $\Delta\omega \to 0$ as $N \to \infty$. The $\phi_0, \phi_1, \phi_2, \ldots, \phi_{N-1}$ are in dependent random phase angles and distributed uniformly in the range of $[0, 2\pi]$.

4. Universal Approximation Theorem

It is aimed at mapping the inputs to output using ANN. This can be accomplished based on the universal approximation theorem which is stated as follows:

Let $\varphi(\cdot)$ be a nonconstant, bounded, and monotone increasing, continuous function. Let $I(m)$ denoted m_0 dimensional unit hypercube $[0,1]^{m_0}$. The space of continuous function on I_{m_0} is denoted by $C\left(I_{m_0}\right)$ and $\varepsilon > 0$, there exist an integer m_1 and set of real constants α_1, b_i , and w_{ij}, where $i = 1, \ldots, m_1$ and $j = 1, \ldots, m_0$ such that

$$F\left(x_1, \ldots, x_{m_0}\right) = \sum_{i=1}^{m_1} \alpha_i \varphi\left(\sum_{j=1}^{m_0} w_{ij} x_j + b_i\right) \tag{4.1}$$

is defined as an approximate realization of the function $f\left(x_1, \ldots, x_{m_0}\right)$ for all $x_1, x_2, ..., x_{m_0}$ that lie in the input space. The universal approximation theorem is directly applicable to multilayer perceptrons As a case, the hyperbolic tangent function that is used in a neuron model is indeed a nonconstant, bounded and monotone-increasing function. Consequently, ANN satisfies conditions imposed on the function $\varphi(\cdot)$

The universal approximation theorem is an existence theorem. The theorem provides the mathematical justification for the approximation of an arbitrary continuous function as opposed to exact representation. Equation (2.12), which is the essence of the theorem, merely generalizes approximations by finite Fourier series. In effect, the theorem states that a single hidden layer is sufficient for a multilayer perceptron to compute a uniform ε approximation to a given training set represented by the set of input x_{m_0} and a desired (target) output, x_{m_0}.

5. Application of ANN as Surrogate Model

Consider a Multi-degree of Freedom (MDOF) linear dynamic system subject to non-stationary excitation. In particular, the non-stationary excitation is a stochastic non-stationary ground excitation $A_g(t)$. Based on the universal approximation theorem, an ANN will be used as a mapping function from the stochastic non-stationary ground excitation from time to t_0 in order to predict the dynamic response at to e.g. the displacement $X\left(t_0\right)$, i.e.

$$X\left(t_n\right) = ANN\left(A_g\left(t_0\right), \ldots, A_g\left(t_n\right)\right) \tag{5.1}$$

where ANN is the ANN model acting as a mapping function from the inputs $\{A_g(t_0)$, $\ldots$, $A_g(t_n)\}$ to the output $X\left(t_n\right)$. The ANN as a surrogate model is obtained from the following steps.

1. Define the architecture of ANN. In this study, a three-layer feed-forward ANN is used. There are input, hidden, and output layers. The number of hidden layer is limited to one. Let the number of neurons in the hidden layer be equal to N_h.

2. Generate N numbers of stochastic ground excitation according to the procedure in Section 3.

3. Input the N generated excitation to the Finite Element Model (FEM) to compute the desired dynamic responses. The total number of responses is also equal to N.

4. Divide the N generated excitations and their corresponding responses into two groups. The first group consists of N_{train} realizations of generated excitations and their

corresponding responses. This group is denoted as the training set. The testing group consists of N_{test} generated excitations and their corresponding response. Note that $N_{\text{train}} + N_{\text{test}} = N$.

5. Employ the N_{train} excitations and their FEM dynamic responses as the input and output of the ANN in order to train the ANN using the back-propagation algorithm.

6. Input the N_{test} excitations to the trained ANN and compute the N_{test} ANN-based dynamic responses.

7. Compute the relative error ε from

$$\varepsilon = \frac{1}{N_{\text{test}}} \sum_{i=1}^{N_{\text{test}}} \left| \frac{X_{FEM,i} - X_{ANN,i}}{X_{ANN,i}} \right|. \tag{5.2}$$

8. Vary the number of N_h and select the N_h with smallest ε.

The application of ANN as surrogate model will be shown in the next section.

6. Illustrative Example

Consider a 3-DOF linear system of shear frame building. The building subjects to the non-stationary excitation. The equation of motion is given by

$$\mathbf{M}\ddot{\mathbf{X}} + \mathbf{C}\dot{\mathbf{X}} + \mathbf{K}\mathbf{X} = -A_g\mathbf{M}\boldsymbol{\Gamma} \tag{6.1}$$

in which $\mathbf{M}, \mathbf{C}, \mathbf{K}$ is the mass, damping, and stiffness matrix, respectively. $\ddot{\mathbf{X}}, \dot{\mathbf{X}}, \mathbf{X}$ are the acceleration, velocity, and displacement matrix, respectively. While as A_g is the non-stationary ground excitation. The matrix term $\boldsymbol{\Gamma}$ is defined by

$$\boldsymbol{\Gamma} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}. \tag{6.2}$$

The following parameters are used in the study.

$$M = \begin{bmatrix} 100 & 0 & 0 \\ 0 & 100 & 0 \\ 0 & 0 & 100 \end{bmatrix} \tag{6.3}$$

$$C = \begin{bmatrix} 2500 & -2500 & 0 \\ -2500 & 3500 & -1000 \\ 0 & -1000 & 3000 \end{bmatrix} \tag{6.4}$$

$$K = \begin{bmatrix} 1.0e7 & -1.0e7 & 0 \\ -1.0e7 & 1.5e7 & -0.5e7 \\ 0 & -0.5e7 & 1.0e7 \end{bmatrix} \tag{6.5}$$

where $S_0 = 0.001$, $\omega_g = 15$, $\eta_g = 0.25$, $A_0 = 2.87$, $\alpha = 0.13$, $\beta = 0.35$, $r = 1.0$, $\omega_m = 1000$, $t_m = 5$, and $\omega_u = 1000$.

1000 realizations of A_g are generated according to the prescribed procedure in Section 3 and parameters above. Each realization of A_g starts from $t = 0$ to $t = 15$ s, with the time step size or discretization $dt = 0.05$. Specifically, $X_1(15), X_2(15)$, and $X_3(15)$ are of interest. A generated realization is shown in Figure 4.

700 realizations of A_g are used for training ANN and therefore are input in the FEM model to compute the 700 realized responses $X_1(15), X_2(15)$, and $X_3(15)$. The 700 realizations of A_g and respective response $X_i(15) (i = 1, 2, 3)$ are used as input and output of

ANN for its training. The generic architecture of ANN is given in Figure 5. Accordingly, each response prediction has its own ANN architecture.

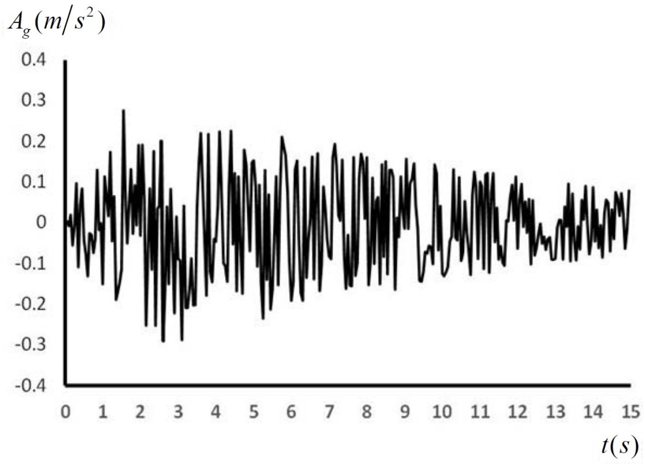

FIGURE 4. A realization of A_g.

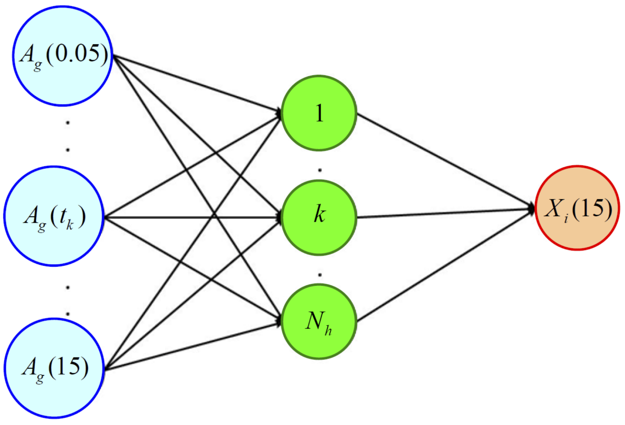

FIGURE 5. Generic architecture of ANN surrogate model for prediction of $X_i (i = 1, 2, 3)$.

The remaining 200 realizations of A_g are input to the FEM model and the trained ANN and computes the respective responses $X_1(15), X_2(15)$, and $X_3(15)$, respectively. These 200 realization is the test samples. Both FEM and ANN-based results using N_h neurons in the hidden layer that yields the least relative error ε for each response are reported in Table 1 . The ANN having N_h with the least ε is referred to as the best ANN.

The best ANN for each response is then employed for the stochastic analysis using Monte Carlo Simulation (MCS). In this stochastic analysis, the uncertainty in soil properties is considered while the other parameters remain the same as those above.

TABLE 1. N_h with the least relative error ε for each response $X_i(15)$ from test samples.

Response	N_h	ε
$X_1(15)$	10	0.000461
$X_2(15)$	8	0.000064
$X_3(15)$	5	0.000152

The Probability Density Function (PDF) of each soil parameter is given as

$$\omega_g \sim \text{lognormal}\left(2.703, 9.975 \times 10^{-2}\right) \tag{6.6}$$

$$\eta_g \sim \text{lognormal}\left(-1.391, 9.975 \times 10^{-2}\right) \tag{6.7}$$

MCS is performed with 1000 realizations of A_g according to the both PDFs defined in (2.20) and (2.21). All realizations of A_g are input to the FEM and the respective best ANN model from Table 1. 1000 responses of $X_1(15), X_2(15)$, and $X_3(15)$ are computed from both FEM and best ANN models. The results are compared in Figure 6 to Figure 8. The accuracy of the ANN in predicting the stochastic responses is shown in Table 2.

TABLE 2. Accuracy of the ANN in predicting the stochastic response.

Response	Relative error in prediction
$X_1(15)$	0.00409
$X_2(15)$	0.00337
$X_3(15)$	0.00430

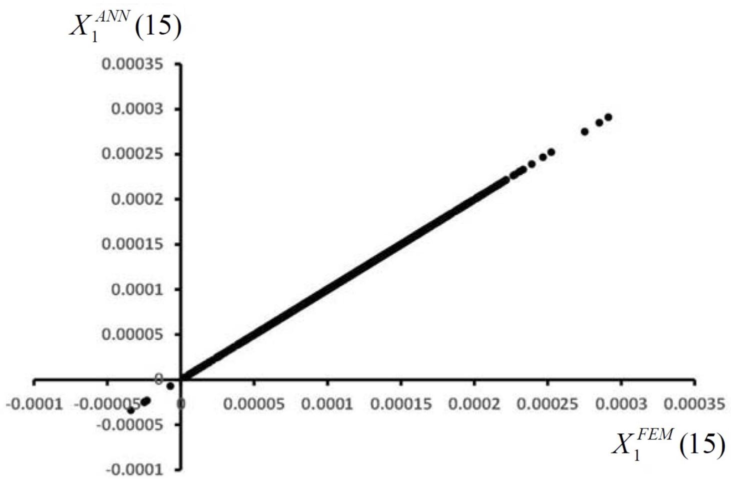

FIGURE 6. Comparison of $X_1(15)$ from the FEM and the best ANN model.

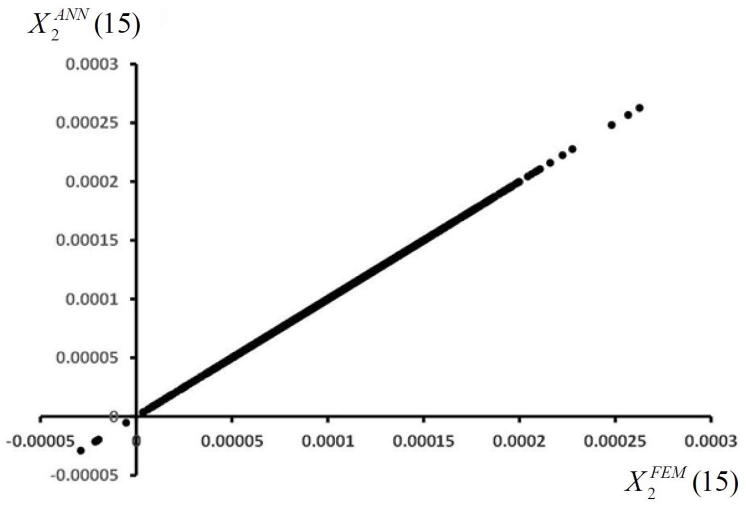

FIGURE 7. Comparison of $X_2(15)$ from the FEM and the best ANN model.

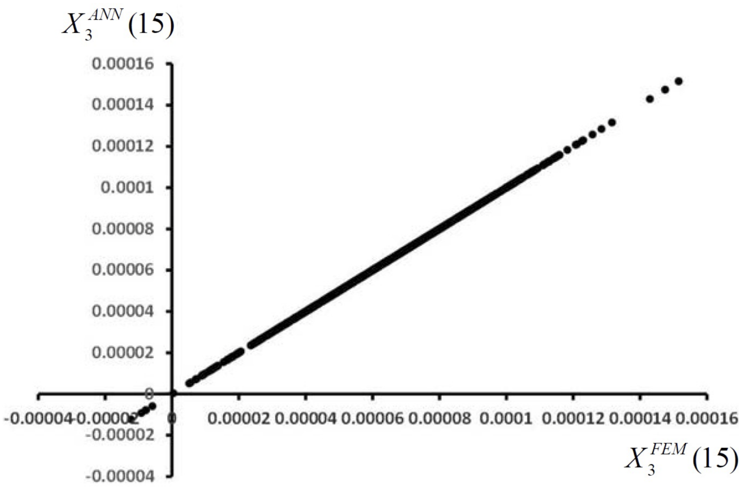

FIGURE 8. Comparison of $X_3(15)$ from the FEM and the best ANN model.

Figure 6 to Figure 8 as well as the relative errors in prediction indicate that the ANN model performs distinctly well as the surrogate models for time-domain analysis of linear dynamic systems subject to non-stationary excitations.

The computational time of the used FEM with Newmark method [26], which is a standard and conventional tool for dynamic analysis, is compared with the proposed ANN surrogate model. The results are given in Table 3. The computing machine has 8 GB ram with 9th Gen Intel $^{\circledR}$core$^{\text{TM}}$i7. The proposed ANN model is significantly more efficient than the standard FEM 30-200 times in terms of computational effort (confer Table 3). This gain of efficiency is due to the forward and one-way computation of ANN whereas the

Newmark method uses the recursive scheme. The comparison of the computational efforts informs that ANN is significantly superior to FEM in terms of computational efficiency.

Table 3. Comparison of computational times between the FEM and the proposed ANN model.

Response	Time used by FEM (Seconds)	Time used by ANN (Seconds)	Computation Efficiency of ANN over FEM (times)
X1(15)	14.779212	0.480077	30.78508656
X2(15)	14.024838	0.082745	169.4946885
X3(15)	14.018782	0.068174	205.6323818

7. Conclusions

Feed-forward artificial neural net-works (ANNs) are proposed as a surrogate model for time-domain analysis of linear dynamic systems subject to non-stationary excitations. The model is of the multi-layer type that contains at least one hidden layer. The back-propagation is employed for the learning of the ANNs. The realizations of non-stationary excitations are simulated according to the evolutionary power spectral density function model using time series presentation. A 3-DOF linear system subjecting to a non-stationary stochastic ground motion due to random soil proper-ties is used as an illustrative example. The numerical results show the ANN performs excellent as the surrogate model. The ANN models not only yields accurate results but is also significantly superior to the analysis using Finite Element Method in terms of computational efficiency. The proposed methodology has the potential of application for the risk and reliability analysis of the system with uncertain parameters. The system includes civil engineering structures subject to earthquakes, e.g. buildings and dams, the vibration of aerospace structures, the offshore platforms subjected to dynamic waves, etc. The other practical and realistic applications can be found in [27].

Acknowledgements

The authors would like to thank the referees for their comments and suggestions on the manuscript.

References

[1] N. Harnpornchai, N. Chakpitak, T. Chandarasupsang, T.A. Chaikijkosi, K. Dahal, Dynamic adjustment of age distribution in Human Resource Management by genetic algorithms, In 2007 IEEE Congress on Evolutionary Computation (2007) 1234–1239.

[2] P. Paokanta, M. Ceccarelli, N. Harnpornchai, N. Chakpitak, S. Srichairatanakool, Rule induction for screening Thalassemia using machine learning techniques: C5.0 and CART, ICIC Express Letters 6 (2) (2012) 301–306.

[3] P. Paokanta, N. Harnpornchai, Risk analysis of Thalassemia using knowledge representation model: Diagnostic Bayesian Networks, Proceedings - IEEE-EMBS International Conference on Biomedical and Health Informatics: Global Grand Challenge of Health Informatics, BHI 2012 (2012) 155–158.

[4] P. Paokanta, N. Harnpornchai, N. Chakpitak, S. Srichairatanakool, M. Ceccarelli, Knowledge and data engineering: Fuzzy approach and genetic algorithms for clustering β-Thalassemia of knowledge based diagnosis decision support system, ICIC Express Letters 7 (2) (2013) 479–484.

[5] P. Paokanta, N. Harnpornchai, N. Chakpitak, The classification performance of binomial logistic regression based on classical and Bayesian statistics for screening P-Thalassemia, The 3rd International Conference on Data Mining and Intelligent Information Technology Applications. IEEE (2011).

[6] N. Harnpornchai, W. Wonggattaleekam, An Application of Neutrosophic Set to Relative Importance Assignment in AHP, Mathematics 9 (2021) 2636.

[7] N. Harnpornchai, W. Wonggattaleekam, A Nikaido Isoda-based hybrid genetic algorithm and relaxation method for finding Nash equilibrium, Mathematics 10 (2022) 81.

[8] K. Hornik, Approximation capabilities of multilayer feedforward networks, Neural Networks 4 (1991) 251–257.

[9] L. Schueremans, D.V. Gemert, Benefit of splines and neural networks in simulation based structural reliability analysis, Structural Safety 27 (3) (2005) 246–261.

[10] L. Schueremans, Use of Meta-Models in structural reliability-new issues in the applicability of probabilistic techniques for construction technology, Ongoing Postdoctoral Research, KULeuven, url: http://www. kuleuven. ac. be/bwk/materials, Research/index. htm (2004).

[11] S. Shao, T. Murotso, Structural reliability analysis using a neural network, JSME International Journal Series A Solid Mechanics and Material Engineering 40 (3) (1997) 242–246.

[12] T. Sasaki, A neural network-based response surface approach for computing failure probabilities, In: Corotis, RB, Schuëller, GI, Shinozuka, M. (Eds.), The 8th International Conference on Structural Safety and Reliability (ICOSSAR2001), USA (2001).

[13] A.T. Goh, F.H. Kulhawy, Neural network approach to model the limit state surface for reliability analysis, Canadian Geotechnical Journal 40 (6) (2003) 1235–1244.

[14] V. Le, L. Caracoglia, A neural network surrogate model for the performance assessment of a vertical structure subjected to non-stationary, tornadic wind loads. Computers & Structures 231 (2020) 106208.

[15] X. Cheng, W. Shao, K. Wang, B.Z. Wang, An ANN-based surrogate model for wave propagation in uncertain media, Waves in Random and Complex Media (2021) 1–14

[16] A.C. de Pina, A.A. de Pina, C.H. Albrecht, B.S.L.P. de Lima, B.P. Jacob, ANN-based surrogate models for the analysis of mooring lines and risers, Applied Ocean Research 41 (2013) 76–86.

[17] M. Ziyadi, I.L. Al-Qadi, Efficient surrogate method for predicting pavement response to various tire configurations, Neural Computing and Applications 28 (6) (2017) 1355–1367.

[18] H.B. Ly, B.T. Pham, L.M. Le, T.T. Le, V.M. Le, P.G. Asteris, Estimation of axial load-carrying capacity of concrete-filled steel tubes using surrogate models, Neural Computing and Applications 33 (8) (2021) 3437–3458.

[19] P.G. Asteris, K.G. Kolovos, Self-compacting concrete strength prediction using surrogate models, Neural Computing and Applications 31 (1) (2019) 409–424.

[20] D.E. Rumelhart,J.L. McClelland, eds. Parallel Distributed Processing: Explorations in the Microstructure of Cognition, vol.1, Cambridge, MA: MIT Press, 1986.

[21] S. Haykin, Neural Networks and Learning Machines (3rd edition), Pearson Education, Inc., Upper Saddle River, New Jersey, 2009.

[22] K. Kanai, Semi-empirical formula for the seismic characteristics of the ground, Bulletin of the earthquake research institute 35 (1957) 309–325.

[23] H. Tajimi, A statistical method of determining the maximum response of a building during earthquake, Proceedings of World Conference on Earthquake Engineering (1960).

[24] M.B. Priestley, Evolutionary spectra and nonstationary processes, Journal of the Royal Statistical Society: Series B (Methodological) 27 (2) (1965) 204–229.

[25] M.B. Priestley, Power spectral analysis of non-stationary random processes, Journal of Sound and Vibration 6 (1) (1967) 86–97.

[26] N.M. Newmark, A method of computation for structural dynamics, Journal of the Engineering Mechanics Division 85 (3) (1959) 67–94.

[27] D. Tao, J. Lin, Z. Lu, Time-frequency energy distribution of ground motion and its effect on the dynamic response of nonlinear structures, Sustainability 11 (3) (2019) 702.