Hindawi Publishing Corporation
International Journal of Aerospace Engineering
Volume 2011, Article ID 247294, 7 pages
doi:10.1155/2011/247294

*Research Article*

# Efficient Neural Network Modeling for Flight and Space Dynamics Simulation

**Ayman Hamdy Kassem**

*Aerospace Engineering Department, Cairo University, Cairo 12613, Egypt*

Correspondence should be addressed to Ayman Hamdy Kassem, draymank@yahoo.com

Received 26 March 2011; Revised 19 July 2011; Accepted 9 August 2011

Academic Editor: C. B. Allen



This paper represents an efficient technique for neural network modeling of flight and space dynamics simulation. The technique will free the neural network designer from guessing the size and structure for the required neural network model and will help to minimize the number of neurons. For linear flight/space dynamics systems, the technique can find the network weights and biases directly by solving a system of linear equations without the need for training. Nonlinear flight dynamic systems can be easily modeled by training its linearized models keeping the same network structure. The training is fast, as it uses the linear system knowledge to speed up the training process. The technique is tested on different flight/space dynamic models and showed promising results.

## 1. Introduction

An artificial neural network (ANN), usually abbreviated as neural network (NN), is a mathematical model that is inspired by the structure of the brain biological neural networks. A neural network consists of an interconnected group of elements called neurons, arranged in different layers, and it processes information by propagating data through these elements. Neural networks use "weights" to change the parameters of the connections to the neurons. These weights are calculated through a learning process of the available data. Simply speaking, neural networks are nonlinear statistical data modeling tools used to model complex relationships between inputs and outputs. Two challenges face neural network designer: the selection of neural network topology (i.e., number of layers and number of neurons) and the initialization of the weights. It is well recognized that a perfect way to decide an appropriate architecture and assign initial values to start neural network training has yet to be established. A common criticism of neural networks came from those two challenges where wrong selection of topology or initial values for weights leads to slow operation [1–5].

A lot of work has already been devoted to the application of neural networks (NN) in dynamics and control, both with respect to controller design and to system modeling and identification [6–10]. Modeling dynamic systems or controllers by neural networks consumes a lot of the network designer time in selecting the appropriate structure and in training the network and had to be done offline. The network size also affects the execution time and has high impact on network design and selection especially for real-time problems.

The network design technique given in this paper is systematic to relieve the network designer from the intensive task of finding an appropriate structure for the neural model. For linear systems, based on the work done by Kaiser [11] and Kassem and Sameh [12], the paper can find the network weights and biases analytically so that the neural network can work directly without the need for training.

Nonlinear systems can be modeled by training its linearized models about specific operating point without the need to change the size or the structure of the network. The training is fast and can be done online, as it uses the linear system knowledge to speed up the training process [12].

A modified version of time-delay neural network is presented in this work, and when it is combined with the analytical modeling, it can fast and accurately model linear and nonlinear dynamic systems. The technique is tested on different flight and space dynamics models and showed good match with the linear and nonlinear equations.

## 2. NN Analytical Model with One Hidden Neuron

This section represents the mathematical equations for linearizing a neural network model consisting of an input neuron, a hidden neuron, and an output neuron (Figure 1). The input and output neurons have linear activation functions, and the hidden neuron has a sigmoid function. The ability of this neural network model to approximate the straight-line equation $y = ax + b$ is an essential feature to allow direct representation of difference equations. During the following calculations, $\sigma(x)$ denotes a continuous nonlinear transfer function applied in the neurons such as the sigmoid function $\sigma(x) = 1/(1 + e^{-x})$.

Assuming that $y = ax + b$ is to be approximated for $x \in [x_0, x_1]$, $x_1 > x_0$ within an error limit of $\varepsilon$, $\varepsilon > 0$, select $z_0$, $z_1$ on the sigmoid horizontal axis with $z_1 > z_0$ such that $|\dot{\sigma}(z) - qa| < \varepsilon/|x_1 - x_0|$ for $z \in [z_0, z_1]$ with a constant $q \in R$. In other words, this step selects the interval on the horizontal axis of the sigmoid, where the sigmoid function can be seen as linear.

Define a function $f$, $f : [x_0, x_1] \to [z_0, z_1]$ as

$$\begin{aligned} f(x) &= z_0 + \frac{z_1 - z_0}{x_1 - x_0}(x - x_0) \\ &= \left(z_0 - \frac{z_1 - z_0}{x_1 - x_0}x_0\right) + \frac{z_1 - z_0}{x_1 - x_0}x \\ &= p_1 + p_2 x. \end{aligned} \tag{1}$$

Finally, further modifications (see Kaiser 1994 [11] for details) yield

$$ax + b = w(p_2 x + p_1) + w_0, \tag{2}$$

where

$$w = \frac{1}{p_2 q}, \qquad w_0 = \frac{-ap_1}{p_2} - \frac{\sigma(0)}{p_2 q} + b. \tag{3}$$

Taking $p_2$ as the weight to the corresponding input unit, $p_1$ as the bias of the hidden neuron, $w$ as the weight of the link between the hidden neuron and an output unit, and $w_0$ as the bias of the output unit results in the desired approximation. This analytic approximation forms the base for the algorithm that will be used to model the dynamics of linear systems. Figure 1 shows the model used in this paper for approximating the linear function.

## 3. Modified Time-Delay Neural Networks

Time-delay neural networks (TDNNs), introduced by Waibel et al. [13], form a class of neural networks with a special topology. They are used for position-independent recognition of features within a larger pattern. In order to be able to recognize patterns place or time invariant, older activation and connection values of the feature units have to be stored. This is performed by making a copy of the feature units with all their outgoing connections in each time step before updating the original units. The total number of time steps saved by this procedure is called delay.

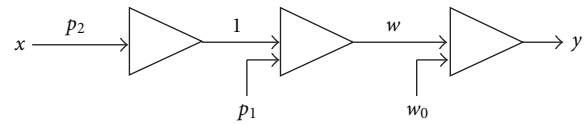


Figure 1: The model used for approximating the function $y = ax + b$.

In this work, the TDNN is modified to fit exactly the general difference equation for a linear $n$th order dynamic system with constant coefficients shown as follows [14]:

$$\begin{aligned} &y(n+k) + a_{n-1}y(n+k-1) + \\ &\quad \cdots + a_1 y(k+1) + a_0 y(k) = f(k), \end{aligned} \tag{4}$$

where $y(i), i = k, k+1, \ldots, k+2$ denotes the discrete variable $y$ at the $i$th instant and $f(k)$ denotes the input value which could be another difference equation.

The output $y(n + k)$ is separated to one side of the equation, while its delayed values and the input are moved to the other side of the equation as follows:

$$\begin{aligned} &y(n+k) \\ &= -a_{n-1}y(n+k-1) - \cdots - a_1 y(k+1) - a_0 y(k) + f(k). \end{aligned} \tag{5}$$

Now, the output value is just a summation of linear relations in the form $y = ax$ which can be modeled analytically as an NN with one hidden neuron. This way, the one hidden neuron NN is used as a building block, and the complete network is built systematically as shown in Figure 2.

## 4. The Algorithm

First, the nonlinear system is linearized about an operating point (trim condition) using any linearization technique such as Taylor series or small disturbance theory. Second, an analytic NN model is build using this linear model after discretization. Finally, the model is trained with the nonlinear model data points and with initial values for weights and biases taken from the linear analytic NN model. The detailed modeling algorithm is shown in Algorithm 1. For linear systems steps 2, 3, and 4 only apply, as there is no linearization, and there is no need for training.

## 5. Test Cases

Three test cases will be given in this section illustrating the application of the proposed modeling approach using

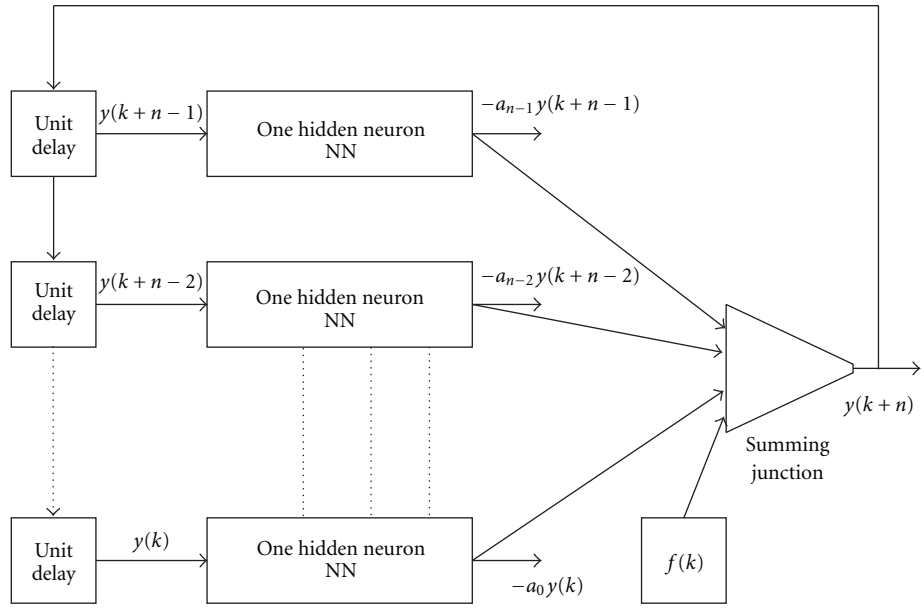


Figure 2: NN for modeling difference equation.

(1) Starting with the nonlinear model in the form of
$\dot{x}(t) = f(x, u, t)$,
where $x$ are states vector and $u$ is input vector.
(2) Linearize the nonlinear model around an operating point and represent it in state space form as follows:
$\dot{x}(t) = Ax + Bu$,
where $A$, $B$ are the system and input matrices, respectively.
(3) Discretize the continuous linearized system using finite difference, the equations will be
$\dot{x}_{i+1} = (A + I\Delta t)x_i + B\Delta t u_i$,
where $I$ is the identity matrix and $\Delta t$ is the time step.
(4) Build an analytic NN to replicate the linear model using the proposed linear system given in section two and Figure 2.
(5) Train the linear analytic NN with the nonlinear model data and with initial weights and biases taken from the linear analytic NN.
(6) If the final NN model is not satisfactory, then increment model size by adding another linear model at different operating point, repeat steps 2 to 5, and concatenate the two models to get a better approximation. The number of final concatenated models depends on the required accuracy and the acceptable execution time.
(7) Stop

Algorithm 1: NN algorithm for modeling nonlinear systems.

NN. The first two test cases in Sections 5.1 and 5.2 show the result of modeling of unstable linear dynamics without details of the proposed algorithm. The third nonlinear test case presented in Section 5.3 will be accompanied with more details.

*5.1. Helicopter Longitudinal Dynamics.* The state space linear model of longitudinal motion of a helicopter [15] is given by

$$\begin{bmatrix} \dot{q} \\ \dot{\theta} \\ \dot{u} \end{bmatrix} = \begin{bmatrix} -0.4 & 0 & -0.01 \\ 1 & 0 & 0 \\ -1.4 & 9.8 & -0.02 \end{bmatrix} \begin{bmatrix} q \\ \theta \\ u \end{bmatrix} + \begin{bmatrix} 6.3 \\ 0 \\ 9.8 \end{bmatrix} \delta, \quad (6)$$

where $q$, $\theta$, and $v$ are pitch rate, pitch angle of the fuselage, and horizontal forward velocity, respectively, and the control input $\delta$ is the rotor tilt angle. The system is unstable with both zeros and poles in the right hand plane, for example, system poles

$$p_{1,2} = 0.3776 \pm 0.3445j; \qquad p_3 = -0.375. \quad (7)$$

Using the linear algorithm, the NN can fit the system's three states very accurately for unit step $\delta$ input as shown in Figure 3.

*5.2. F-16 Longitudinal Dynamics.* The equations for longitudinal motion of the F-16 fighter are considered for a trimmed flight condition with $U = 502$ ft/s, $\alpha = 0.0393$ rad, $\theta = 0.0393$ rad, and $q = 0$ rad/s. The state vector is $x = [\Delta u, \Delta\alpha, \Delta\theta, \Delta q]$, where $\Delta v$, $\Delta\alpha$, $\Delta\theta$, and $\Delta q$ are the longitudinal velocity, angle of attack, pitch angle, and pitch

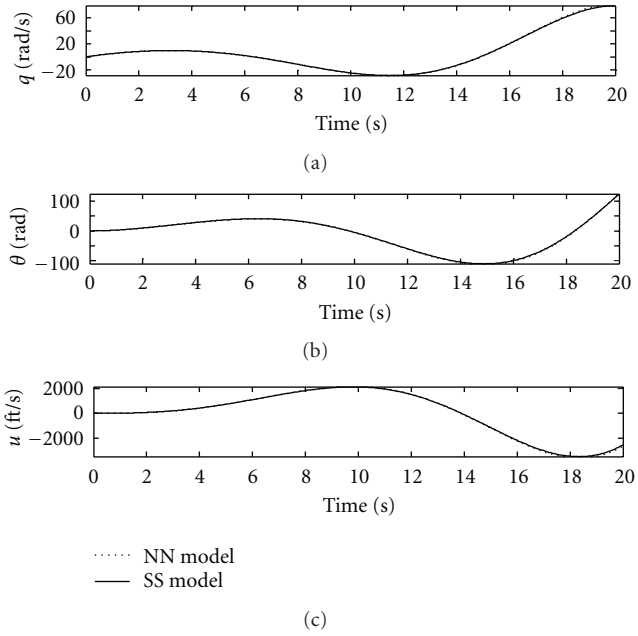


Figure 3: NN versus state-space models for helicopter dynamics described by (6).

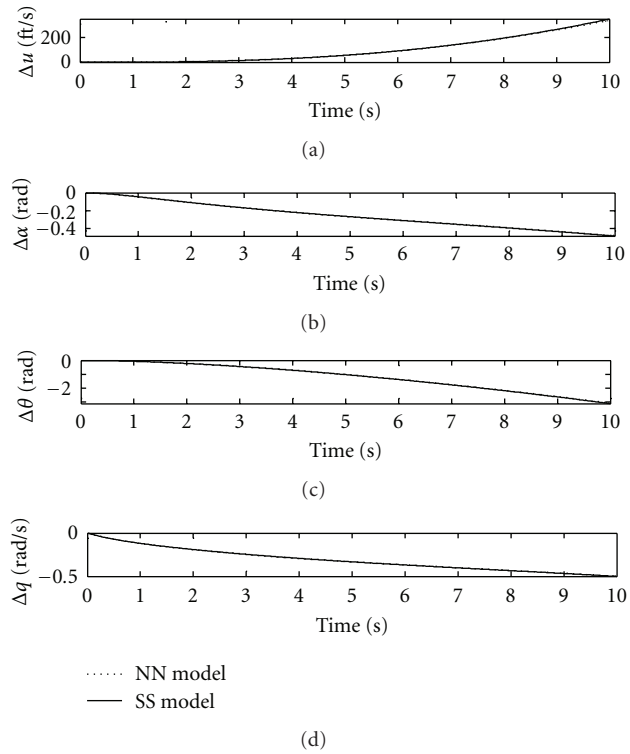


Figure 4: NN versus state-space models for F-16 longitudinal dynamics.

rate, respectively. The single control input is the elevator deflection $\delta_E$ in degrees. The state equation in matrix form becomes [16]

$$\dot{x} = Ax + B\delta_E, \tag{8}$$

where

$$A = \begin{bmatrix} -1.9311e-2 & 8.8157 & -32.17 & -0.575 \\ -2.5389e-4 & -1.0189 & 0 & 0.9051 \\ 0 & 0 & 0 & 1 \\ 2.9465e-12 & 0.8223 & 0 & -1.0774 \end{bmatrix},$$
$$B = \begin{bmatrix} 0.1737 \\ -2.1499 \\ 0 \\ -0.1756 \end{bmatrix}. \tag{9}$$

Using the linear algorithm, the NN can fit the system's four states very accurately for unit step input $\delta_E$, as shown in Figure 4.

*5.3. Generic Fighter Nonlinear Flight Dynamics.* Consider the flight dynamics model of a high-performance aircraft for the short-period mode [17–19], where the motion involves rapid changes to the angle of attack and pitch attitude at roughly constant airspeed. The model covers both linear and nonlinear flight behavior through an extensive range in angle of attack. The model is mathematically described as [18]

$$\dot{\alpha} = q + 9.168C_z - 1.834(\delta_e + 7) + 7.362,$$
$$\dot{q} = 5.73(-\alpha - 1.5\delta_E) + 2.865, \tag{10}$$

where

$$C_z = \begin{cases} -0.07378\alpha & \alpha \le 14.36, \\ 0.09722\alpha^2 - 2.865\alpha + 20.04 & 14.36 < \alpha \le 15.6, \\ -0.01971\alpha^2 + 0.7439\alpha - 7.808 & 15.6 < \alpha < 19.6, \\ -0.4733 - 0.01667\alpha & \alpha \le 19.6, \end{cases} \tag{11}$$

and $\alpha$ is the angle of attack in degrees, $q$ is the pitch rate in degrees per second, $\delta_e$ is the elevator control surface deflection in degrees, and $C_Z(\alpha)$ is the normal force coefficient and the sole nonlinearity in the model. The system dynamics can be summarized as

(a) with angle of attack <14.36, the aircraft has stable dynamics,

(b) with angle of attack between 14.36 and 19.6 degrees, the aircraft has oscillatory instability,

(c) onset of limit cycle oscillations (LCO) at high angles of attack.

It is obvious that a single linear model cannot represent this nonlinear behavior. So, this example will show the strength of the proposed algorithm. It will start with a single linear NN model and then train it to capture the nonlinear behavior, then adding a second model at different operating point to improve the accuracy.

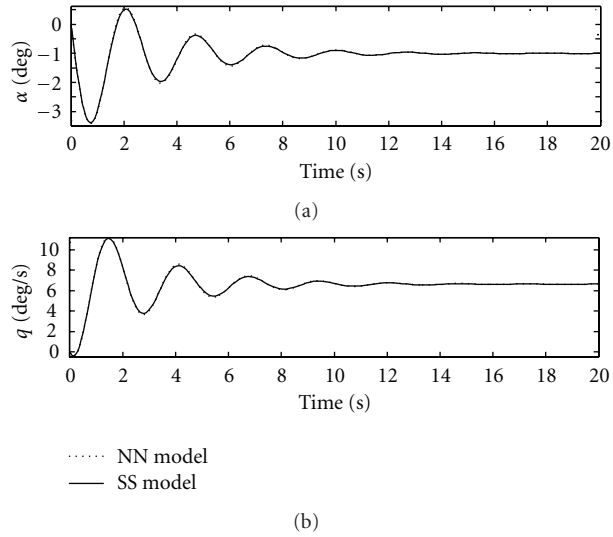


Figure 5: Linear NN model versus state-space model for fighter linearized dynamics.

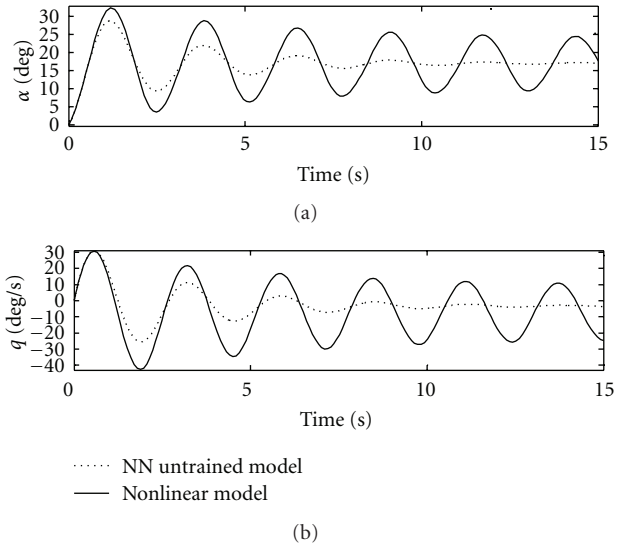


Figure 6: NN trained model versus nonlinear models for fighter dynamics.

The linearized model for $\alpha < 14.36$ deg is given by

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.767 & 1 \\ -5.73 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -1.834 \\ 8.595 \end{bmatrix} \delta_E + \begin{bmatrix} -5.476 \\ 2.865 \end{bmatrix}. \tag{12}$$

The discretized system (12) using $\Delta t = 0.004$ s is given by

$$\begin{bmatrix} \alpha_{i+1} \\ q_{i+1} \end{bmatrix} = \begin{bmatrix} 9.973e-01 & 4.0e-03 \\ -2.292e-02 & 1.00 \end{bmatrix} \begin{bmatrix} \alpha_i \\ q_i \end{bmatrix} + \begin{bmatrix} -7.336e-03 \\ -3.438e-02 \end{bmatrix} \delta_{E_i} + \begin{bmatrix} -2.1904e-02 \\ 1.1460e-02 \end{bmatrix}. \tag{13}$$

Using the linear algorithm, the NN can fit the system's two states very accurately for the unit step input $\delta_E$, as shown in Figure 5. This assumes that the angle of attack does not exceed value 14.6 deg.

We can see the difference when using the linear NN model for predicting behavior of the nonlinear system when angle of attack exceeds value of 14.6 deg., as shown in Figure 6. Therefore, the NN needs some training. A step input of $\delta_E = -11$ for five seconds is used to train the system, and a fifteen seconds step input with same value ($\delta_E = -11$) is used for testing, and the results are shown in Figure 7. Initial weights and biases were taken from the linear NN to speed up the convergence.

When initiating the NN learning with zero or random weights and biases, it has been noticed that it does not converge to the correct values or in best case converges after 5 times the required number of iterations compared with the case of using linear NN weights and biases.

It is clear that the NN works relatively good compared with its size (4 sigmoid elements). The model can be improved by adding another model (at different operating points).

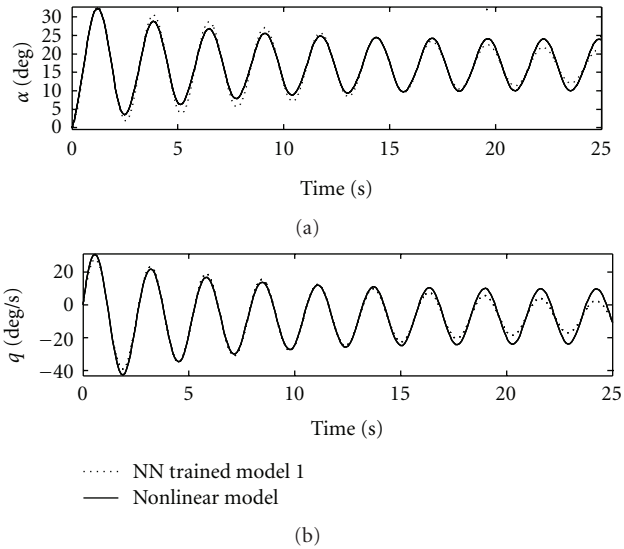


Figure 7: NN trained model 1 versus nonlinear model for fighter dynamics.

Linearizing the system for $\alpha > 19.6$, we get the following state space model:

$$\begin{bmatrix} \dot{\alpha} \\ \dot{q} \end{bmatrix} = \begin{bmatrix} -0.153 & 1 \\ -5.73 & 0 \end{bmatrix} \begin{bmatrix} \alpha \\ q \end{bmatrix} + \begin{bmatrix} -1.834 \\ 8.595 \end{bmatrix} \delta_E + \begin{bmatrix} 8.715 \\ 2.865 \end{bmatrix}. \tag{14}$$

Equation (14) is discretized using $\Delta t = 0.004$ s to give

$$\begin{bmatrix} \alpha_{i+1} \\ q_{i+1} \end{bmatrix} = \begin{bmatrix} 9.994e-01 & 4.0e-03 \\ -2.292e-02 & 1.00 \end{bmatrix} \begin{bmatrix} \alpha_i \\ q_i \end{bmatrix} + \begin{bmatrix} -7.336e-03 \\ -3.438e-02 \end{bmatrix} \delta_{E_i} + \begin{bmatrix} 3.486e-02 \\ 1.146e-02 \end{bmatrix}. \tag{15}$$

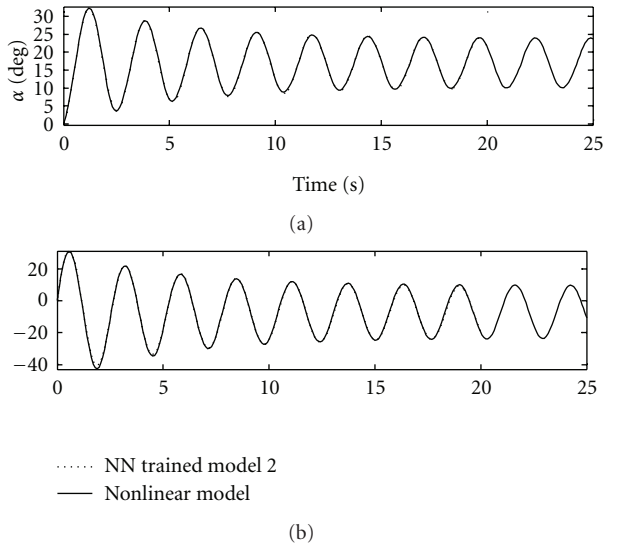


Figure 8: NN trained model 2 versus nonlinear model for generic fighter dynamics.

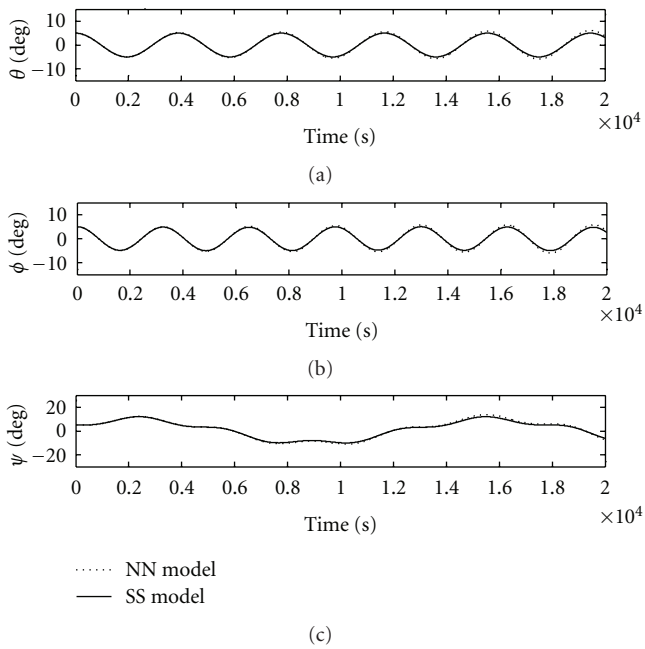


Figure 9: NN versus state-space models for satellite dynamics.

Same procedure is used to produce the NN model for this linear system. Adding the two NN models, we get double the size NN (8 sigmoid elements). After training the concatenating model, we can get an improved model which produces a much better results as shown in Figure 8. It is clear that this improved model can catch the limit cycle very well, so there is no need to add another NN model in the region between 14.36 and 19.6 degrees.

*5.4. Satellite Attitude Dynamics.* This final case study shows a space-related problem and also illustrates the effectiveness of the proposed NN for systems with sparse dynamic matrix ($A$). In this case study, the linearized attitude dynamics for a symmetric satellite for small angles and gravity gradient torques approximation is given by [20]

$$\dot{x} = Ax + Bu, \tag{16}$$

where

$$x = \begin{bmatrix} \theta \\ \phi \\ \psi \\ \dot{\theta} \\ \dot{\phi} \\ \dot{\psi} \end{bmatrix}, \quad A = \begin{bmatrix} 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 \\ \delta_1 & 0 & 0 & 0 & 0 & 0 \\ 0 & \delta_2 & 0 & 0 & 0 & \delta_3 \\ 0 & 0 & \delta_4 & 0 & \delta_5 & 0 \end{bmatrix},$$

$$B = \begin{bmatrix} 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & 0 & 0 \\ 0 & \frac{1}{I_y} & 0 \\ \frac{1}{I_x} & 0 & 0 \\ 0 & 0 & \frac{1}{I_z} \end{bmatrix}, \quad u = \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix}, \tag{17}$$

$$\delta_1 = -\frac{3\omega_0^2}{I_y}(I_x - I_z), \quad \delta_2 = -4\frac{\omega_0^2}{I_x}\left(I_y - I_z\right),$$

$$\delta_3 = -\frac{\omega_0}{I_x}\left(I_y - I_z - I_x\right), \quad \delta_4 = -\frac{\omega_0^2}{I_z}\left(I_y - I_x\right),$$

$$\delta_5 = -\frac{\omega_0}{I_y}\left(I_z + I_x - I_y\right).$$

The satellite altitude = 700 km, its moments of inertia $I_x = 80$, $I_y = 82$, and $I_z = 9$ kg-m$^2$, and an initial attitude is represented by $\theta(0) = 5°$, $\phi(0) = 5°$, and $\psi(0) = 5°$. Figure 9 shows initial condition response of the satellite using the state-space and NN models. The NN model is built by direct substitution without the need to train the network, and the results show good match. Also, the NN model is very efficient, as it uses only eight (one hidden neuron NN) for matrix A representing only the nonzero elements.

## 6. Conclusions

The procedure and the test cases described in this paper show that the linear neural network based on the analytic one hidden neuron NN is best suited as a starting point for fast learning nonlinear networks. It also shows that the modified Time-delay neural network combined with the analytical network form a very efficient algorithm for flight dynamics equations approximations. The presented algorithm relieves the network designer from the work intensive task of finding a suitable structure for the neural network, given that the designer has at least some qualitative knowledge

about the modeling task. It also shows a systematic way of increasing the size of the neural network to improve the accuracy without the need of guessing. It is clear that the algorithm gives some guidance to the learning process and helps the network escaping local minima by using initial guess from the linear model. This illustrates the importance and usefulness of this algorithm in the field of neural network modeling of nonlinear flight dynamics.

## References


[1] M. T. Hagan, H. B. Demuth, and M. Beale, *Neural Network Design*, PWS Publishing Company, Boston, Mass, USA, 1995.

[2] K. Hornik, M. Stinchcombe, and H. White, "Multilayer feedforward networks are universal approximators," *Neural Networks*, vol. 2, no. 5, pp. 359–366, 1989.

[3] L. K. Jones, "Constructive approximations for neural networks by sigmoidal functions," *Proceedings of the IEEE*, vol. 78, no. 10, pp. 1586–1589, 1990.

[4] T. L. Burrows and M. Niranjan, "Feed-forward and recurrent neural networks for system identification," Tech. Rep., Cambridge University Engineering Department, CUED/F-INFENG/TR158, 1993.

[5] S. S. Haykin, *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 2nd edition, 1998.

[6] M. Norgaard, O. Ravn, K. N. Poulsen, and L. K. Hansen, *Neural Networks for Modelling and Control of Dynamic Systems*, Springer, London, UK, 2001.

[7] J. S. Pei and A. W. Smyth, "A new approach to design multilayer feedforward neural network architecture in modeling nonlinear restoring forces," Smart Structures and Materials 2005: Sensors and Smart Structures Technologies for Civil, Mechanical, and Aerospace Systems, pp.345–353, 2005.

[8] Johan A. K. Suykens, Joos P. L. Vandewalle, and B. L. De Moor, *Artificial Neural Networks For Modelling And Control Of Non-Linear Systems*, Springer, New York, NY, USA, 1995.

[9] O. Nelles, *Nonlinear System Identification: From Classical Approaches to Neural Networks and Fuzzy Models*, Springer, New York, NY, USA, 2000.

[10] I. W. Sandberg, J. T. Lo, C. L. Fancourt, J. C. Principe, S. Katagiri, and S. Haykin, *Nonlinear Dynamical Systems: Feedforward Neural Network Perspectives*, Wiley-Interscience, New York, NY, USA, 2001.

[11] M. Kaiser, "Time-delay neural networks for control," in *Proceedings of the 4th International Symposium on Robot Control (SYROCO '94)*, Capri, Italy, 2004.

[12] A. Kassem and A. Sameh, "A Fast technique for modeling and control of dynamic systems," in *Proceedings of the 17th International Conference on Computers and Their Applications (ISCA '02)*, pp. 83–87, July 2002.

[13] A. Waibel, T. Hanazawa, G. Hinton, K. Shikano, and K. J. Lang, "Phoneme recognition using time-delay neural networks," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 37, no. 3, pp. 328–339, 1989.

[14] Benjamin C. Kuo, *Automatic Control Systems*, Prentice-Hall, 1995.

[15] D. Obradovic, "On-line training of recurrent neural networks with continuous topology adaptation," *IEEE Transactions on Neural Networks*, vol. 7, no. 1, pp. 222–228, 1996.

[16] B. L. Stevens and F. L. Lewis, *Aircraft Control and Simulation*, John Wiley & Sons, New York, NY, USA, 1992.

[17] R. R. Mohler, "Nonlinear Stability and Control Study of Highly Maneuverable High Performance Aircraft," NASA OSU-ECERept 91-01, 1991.

[18] A. Kassem, "Efficient neural network modeling for flight dynamics equations," in *Proceedings of the International Conference on Modeling, Simulation & Visualization Methods, (MSV '10)*, Las Vegas, Nev, USA, July 2010.

[19] A. Omran and B. Newman, "Piecewise global volterra nonlinear modeling and characterization for aircraft dynamics," *Journal of Guidance, Control, and Dynamics*, vol. 32, no. 3, pp. 749–759, 2009.

[20] T. Habib, A. Kassem, and G. El-Bayoumi, "GPS-based small satellite position and attitude determination simulator," *Journal of Engineering and Applied Science*, vol. 52, no. 1, pp. 71–87, 2005.