Dynamical System Modeling Via Signal Reduction and Neural Network Simulation

Thomas L. Paez
Experimental Structural Dynamics Department
Sandia National Laboratories
Albuquerque, New Mexico
(505) 844-7052
e-mail: tlpaez@sandia.gov

Norman F. Hunter
Engineering Science and Analysis Division
Los Alamos National Laboratory
Los Alamos, New Mexico
(505) 667-2099
e-mail: hunter@lanl.gov

Sandia is a multiprogram laboratory operated by Sandia Corporation, a Lockheed Martin Company, for the United States Department of Energy under contract DE-AC04-94AL85000.

Many dynamical systems tested in the field and the laboratory display significant nonlinear behavior. Accurate characterization of such systems requires modeling in a nonlinear framework. One construct forming a basis for nonlinear modeling is that of the artificial neural network (ANN). However, when system behavior is complex, the amount of data required to perform training can become unreasonable. We reduce the complexity of information present in system response measurements using decomposition via canonical variate analysis. We describe a method for decomposing system responses, then modeling the components with ANNs. A numerical example is presented, along with conclusions and recommendations.

INTRODUCTION

Artificial neural networks (ANNs) have been shown capable of simulating the responses of many types of systems, among them, structural systems. For example, Paez, O'Gorman, and Tucker, (1997a, 1997b) have demonstrated that structural system outputs can be faithfully reproduced using layered perceptron ANNs and connectionist normalized linear spline ANNs in the autoregressive (or recurrent) framework. The potential advantage of system simulation with ANNs is that relatively accurate simulations can be obtained in efficient, fast models. ANNs are inductive in nature, and therefore, require training data. This means that data from physical experiments or other numerical simulations can be used to train ANNs.

With regard to the practical simulation of structural system response, though, it has been noted that there are substantial difficulties in simulating the motions of large, complex systems; these difficulties arise from several sources. First, complexity in a structural system implies that the system may have many components, that are, perhaps, nonlinear. This implies that the autoregressive mapping of past system states and excitation values to future states may involve numerous input variables. ANN size influences its feed forward run time, and therefore, the efficiency of simulation. But more important, size influences training time and the number of exemplars required to train the ANN. Both training time and the number of exemplars increase rapidly with ANN size. Because of this, means must be sought to model complex systems in terms of simpler components.

There are numerous approaches to the decomposition of complex signals. (See, for example, Paez, O'Gorman, and Tucker, 1997b.) The approach most familiar to structural engineers is linear modal analysis, and this can be easily

DISCLAIMER

extended to the nonlinear regime. There are several methods for executing modal analyses. One method that is very efficient and adaptable to nonlinear applications is canonical variate analysis (CVA). CVA is a method that uses structural input and response waveforms to establish a transformation to state space. It can be used for response prediction, and in this framework it uses information from the past of a time series to predict future time series waveforms. Input waveform selection is based on a generalized singular value decomposition.

In this paper we present a detailed description of the CVA method for structural analysis in which the nonlinear state map required by CVA is embodied in an ANN. The following section is a description of the CVA technique. Next, the multivariate linear spline (MVLS) ANN is derived for the local linear state map. An example is presented, and the results show that the MVLS network operates much faster than the corresponding classical CVA analysis.

CANONICAL VARIATE ANALYSIS (CVA)

There are numerous instances when time series analysis is used to form an empirical model of a dynamic system. Models may be constructed directly (in a single computational step) from the time series using autoregressive moving average (ARMA) or pure ANN models. This form of model can be quite useful, but often hides structure inherent in the system. In contrast to direct application of ARMA or ANN models, where the potential model dimension is potentially quite high, a more robust approach seeks to exploit characteristics present in the data to derive a reduced order model. The reduced order model is then used for response prediction or studies of system structure. A number of approaches have been proposed for constructing reduced order models. Indeed, the optimal reduction strategy is dependent on the data and the desired model form. Canonical variate analysis (CVA) is a robust model reduction technique for time series data. (Larimore , 1983). In the following paragraphs, we develop the equations of CVA.

Canonical Variate Analysis was developed by Hotelling, who developed state models based on linear transformations of measured time series data. CVA has been improved and robust methods have been developed where CVA is used to estimate state models of linear systems from time series data (Larimore, 1983). Reduced order nonlinear models have also been constructed from time series data. Prediction of future time series values, estimation of state rank, and computation of Lyapunov exponents are described by Priestly (Priestly 1980), Tong (Tong 1990), Packard (Packard, 1980), Farmer (Farmer 1988), and Larimore (Larimore, 1991).

Our approach combines the advantages of ANNs and CVA. Initially we assume that the state model form is potentially nonlinear, and construct a number of local linear models using CVA. The idea behind CVA is that past waveforms are selected for their utility in predicting future waveforms, in contrast to the ARMA approach, in which past time series values are used to predict a future time series value. We define the past and the future of a multivariate time series as:

$$\begin{aligned} \boldsymbol{p}(t_0) &= \left\{u(t_0-\tau)\, u(t_0-2\tau) \ldots u(t_0-j\tau)\, y(t_0-\tau)\, y(t_0-2\tau) \ldots y(t_0-k\tau)\right\} \\ \boldsymbol{f}(t_0) &= \left\{y(t_0+\tau)\, y(t_0+2\tau) \ldots y(t_0+k\tau)\right\} \end{aligned} \tag{1}$$

The y's are sampled response time series values (potentially vectors, in the multiple response case) and the u's are sampled time series input values. The subscript on the time value is the time index, i.e., t_0 is the time at index 0. The variable τ is the time increment, j is the number of input lags and k is the number of response lags. Matrices of past and future behavior are constructed from sets of past and future waveforms as follows.

$$\boldsymbol{P} = \begin{bmatrix} u(t_0-\tau)\, u(t_0-2\tau) \ldots u(t_0-j\tau)\, y(t_0-\tau)\, y(t_0-2\tau) \ldots y(t_0-k\tau) \\ u(t_1-\tau)\, u(t_1-2\tau) \ldots u(t_1-j\tau)\, y(t_1-\tau)\, y(t_1-2\tau) \ldots y(t_1-k\tau) \\ \ldots \\ u(t_m-\tau)\, u(t_m-2\tau) \ldots u(t_m-j\tau)\, y(t_m-\tau)\, y(t_m-2\tau) \ldots y(t_m-k\tau) \end{bmatrix}$$

$$F = \begin{bmatrix} y(t_0+\tau)\, y(t_0+2\tau) \ldots y(t_0+k\tau) \\ y(t_1+\tau)\, y(t_1+2\tau) \ldots y(t_1+k\tau) \\ \ldots \\ y(t_m+\tau)\, y(t_m+2\tau) \ldots y(t_m+k\tau) \end{bmatrix} \tag{2}$$

where $t_i, i = 0,...,m$, is a sequence of time values. Minimization of the error in predicting the future f(t) from the past p(t) is accomplished using a series of singular value decompositions on $\boldsymbol{P}$ and $\boldsymbol{F}$, leading to a transformation matrix $\boldsymbol{T}$, which selects from the past the information critical to the prediction of the future (Larimore, 1983). $\boldsymbol{T}$ is formed from

$$\boldsymbol{T} = \boldsymbol{U}^T \left(\boldsymbol{P}^T \boldsymbol{P}\right)^{-1/2} \tag{3}$$

where

$$SVD\left[\left(\boldsymbol{P}^T \boldsymbol{P}\right)^{-1/2} \left(\boldsymbol{P}^T \boldsymbol{F}\right) \left(\boldsymbol{F}^T \boldsymbol{F}\right)^{-1/2}\right] = \boldsymbol{U}\boldsymbol{W}\boldsymbol{V}^T \tag{4}$$

and SVD(.) denotes the singular value decomposition. The selection of an optimal "memory" of the past is described by

$$\boldsymbol{m}(t) = \boldsymbol{T}\boldsymbol{p} \tag{5}$$

Once $\boldsymbol{m}$(t) is evaluated at every time t, the state equations may be obtained in a least squares sense using

$$\begin{aligned} \boldsymbol{m}(i+1) &= \boldsymbol{A}\boldsymbol{m}(i) + \boldsymbol{B}\boldsymbol{u}(i) + w(i) \\ \boldsymbol{y}(i) &= \boldsymbol{C}\boldsymbol{m}(i) + \boldsymbol{D}\boldsymbol{u}(i) + \boldsymbol{E}w(i) + v(i) \end{aligned} \tag{6a,b}$$

In Eq. (6) $\boldsymbol{m(i)}$, $\boldsymbol{u(i)}$,*and* $\boldsymbol{y(i)}$ are known at any time t_i, $w(i)$ *and* $v(i)$ are white noise processes which result from errors in the solution, and $\boldsymbol{A}$, $\boldsymbol{B}$, $\boldsymbol{C}$, $\boldsymbol{D}$, and $\boldsymbol{E}$ are determined using least squares. The term $\boldsymbol{E}w(i)$ allows for possible correlation between the state noise $w(i)$ and measurement noise $v(i)$. The possibility of this correlation is necessary to obtain a minimal order state space representation of the process. This process is described in detail in Larimore (1983).

Dealing with the nonlinear case requires use of potentially nonlinear maps which: (1) Transform the past and future matrices $\boldsymbol{P}$ and $\boldsymbol{F}$ and into the estimated states $\boldsymbol{m}$. (Eq. (5) - the transformation map) (2) Map past states $\boldsymbol{m}(i)$ into future states $\boldsymbol{m}(i+1)$. (Eq. (6a) - the state map) (3) Predict future time series values from state and input values. (Eq. (6b) - the prediction map)

In contrast to global modeling, local modeling uses a specially selected set of past and future waveforms to formulate the $\boldsymbol{P}$ and $\boldsymbol{F}$ matrices in Eq. (2) above. In local modeling, the current past waveform $p(t_r)$ is considered a reference waveform. The cartesian distance between other past waveforms and the reference past is

$$d(r,n) = \left\{\left[y(t_r-\tau) - y(t_n-\tau)\right]^2 + \ldots + \left[y(t_r-k\tau) - y(t_n-k\tau)\right]^2\right\} \tag{7}$$

The quantity $d(r,n)$ is the distance between a waveform at time index n and the reference waveform at time index r. Each past waveform has a distance from the reference waveform. In local modeling, m waveforms that have the smallest distance from the reference waveform are used to formulate the matrices $\boldsymbol{P}$ and $\boldsymbol{F}$ of past and future vectors. Intuitively this means that wave shapes which are similar to the reference waveform are used to formulate the local linear model.

The local linear formulation requires repeated evaluation of equations 2-6 for every time series point p for which a model formulation or prediction of the future f is required. In each case neighbors of the reference past are used to formulate the past and future matrices P and F, the transformation matrix T is computed, and the A, B, C, and D matrices calculated. This approach is computationally tractable and allows lots of change in the form of all three of the maps described: the transformation map, the state map, and the prediction map. Repeated local linear formulation is, however, computationally expensive, especially in finding nearest neighbors from a large set of training data and in the computation of singular value decomposition.

This computational expense, combined with our knowledge of the versatility and robustness of ANNs, motivates us to formulate the three mappings of CVA in ANN form. Our procedure utilizes CVA to compute a representative set of transformation matrices, state maps, and prediction maps. With a representative set of known exemplars spanning the model space, we construct neural networks which accurately embody the mappings. Once these mappings are completed, we can readily formulate future predictions, map past states to future states, and past values to states without encountering the computational overhead of SVD. As a start, we concentrate in this paper on the state map, mapping past states into future states, Eq. (6a).

THE MULTIVARIATE LINEAR SPLINE NETWORK

The multivariate linear spline (MVLS) network is an artificial neural network (ANN) of the radial basis function type. The radial basis function artificial neural network was developed by Moody and Darken (1989). It simulates mappings via the superposition of radial basis functions. It is an accurate local approximator, and it trains rapidly, but has the potential for size difficulties as the dimension of the input space grows. A generalization of the radial basis function ANN is the connectionist normalized linear spline (CNLS) network. This was developed by Jones, et al., (1990), and seeks to simulate a mapping by using radial basis functions in a higher order approximation than the radial basis function network. It also has good local accuracy, and is fast-training.

The MVLS network generalizes the CNLS network to multiple output dimensions, and in the framework of its present implementation, views training in a construct more familiar to engineers than that used for the CNLS network. The objective of the MVLS network is to provide a model framework for the simulation of multiple input/multiple output mappings. The model framework must be trainable using exemplars - examples of the correct input/output behavior of the system being simulated.

To commence development of the MVLS network, we let x be an n-dimensional vector input to the system being modeled, and let $z = g(x)$ be its corresponding m-dimensional vector output. We assume that the function $g(.)$ is single valued and deterministic, but that its form and parameters are unknown. We can approximate the mapping from x to z in a region of the input/output space using the linear form:

$$z \cong y = A(x - c) \tag{8}$$

where y is an approximation to z, c is a vector with the same dimension as x in the vicinity of which the approximation is made, and A is an mxn matrix of constants. (In practice, the vector x may be augmented with a 1, the vector c augmented with a 0, and the matrix A augmented with a column of constants to account for the possibility of an offset in the linear model. The vector and matrix dimensions are modified accordingly.) This approximation could be optimized using least squares or weighted least squares, and it would be accurate in the vicinity of the data used to develop it as long as the behavior of the mapping in the neighborhood is truly linear. This optimization is discussed in the following section. The vector c is the "center" of the local linear approximation, and it is therefore called a center vector, or simply a center.

We can develop similar approximations in other neighborhoods of the input vector space. To account for this, we append the subscript j to the coefficient matrix and the center vector:

$$z \cong y = A_j(x - c_j) \qquad j = 0,...,N-1 \tag{9}$$

where N is the number of regions of approximation.

Having developed local linear approximations of the x to z mapping, we can now combine the local approximations to create an approximate, global map. The approximation takes an input vector x_0 and maps it into y_0, an approximation to $g(x_0)$. To accomplish this approximation, we superimpose several of the linear approximations in a series. We weight each component in the series according to its distance from the input vector x_0. Local linear models that are near x_0 are weighted heavily, and those that are further away are lightly weighted. The series is:

$$\sum_j y_0 w_j = \sum_j A_j(x_0 - c_j) w_j \tag{10}$$

where the w_j are the weights attached to the local linear models. The output vector y_0 on the left side is independent of the index j, so it can be removed from the sum and the equation can be simplified to:

$$y_0 = \frac{\sum_j A_j(x_0 - c_j) w_j}{\sum_j w_j} \tag{11}$$

This is the parametric form of the MVLS network.

We choose the form of the multivariate Gaussian probability density function for the weighting expression. It is known as a radial basis function and has the form:

$$w_j = \exp\left(-\beta \left\| x_0 - c_j \right\|^2\right) \tag{12}$$

The quantity β is a network parameter related to the width of the radial basis functions; it is to be optimized, and this will be discussed later in the section.

The MVLS network is used in feed forward operation by specifying the input vector x_0, evaluating the weights w_j using Eq. (12), substituting the weights and input vector into Eq. (11), and evaluating the output, y_0, in Eq. (11). This output should present an interpolation among the training outputs that corresponds to the input as an interpolation among the training inputs.

Note that the range of the summation index j is not specified in Eq. (11). It is clear that the summation should be carried out over those linear models nearest the input vector x_0. There are several ways to accomplish this. In the present code we use a preestablished number of models (Eq. (9)), and choose the ones nearest in cartesian space to make each prediction.

There are two groups of quantities and one scalar that need to be identified in order to establish the MVLS network as an approximator to the relation $z = g(x)$. These are the linear coefficient matrices A_j, $j = 0,...,N-1$, the centers c_j, $j = 0,...,N-1$, and the radial basis function width parameter β. There are clearly many ways in which the parameters could be identified. We have chosen the following approach.

Assume that there are m exemplars of the system input/output behavior, either measured from the system or computed from another model that is to be simulated. Denote these x_j, z_j, $j = 0,...,m-1$. If the input and output

exemplars have not been normalized, then they are now normalized by subtracting off the mean, and dividing by the standard deviation. We select at random from among the m normalized exemplars m_o (m_o<m) cases to be used in training the MVLS network. The remaining m-m_o cases are used to test the trained MVLS network. (Often, 80 to 90% of the exemplars are used to train the network, and the remaining 10 to 20% are used to test it.) Denote the training exemplars $x_j, z_j, j = 0,...,m_o - 1$. (It is important to emphasize that these are not normally taken in any preestablished order from the initial set of exemplars.)

To establish the centers $c_j, j = 0,...,N-1$, we start by choosing at random from among the input training exemplars $x_j, j = 0,...,m_o - 1$, a set of vectors $v_j, j = 0,...,N-1$. We allow the collection of vectors $v_j, j = 0,...,N-1$, to self organize so that they become representative of the entire set of input training exemplars. The self organization operation is carried out as follows: (0) Specify two convergence parameters $\alpha \in (0,1)$, and $\gamma \in (0,1)$. (1) Start an epoch of self organization by randomly shuffling the set of input training exemplars. (2) For each input exemplar in the shuffled set: (a) Determine which of the vectors v_j is nearest (in cartesian space). (b) Move the vector v_j in a straight line toward the training exemplar, a fraction α of the total distance. (3) After this operation has been repeated for all the training exemplars, diminish α by multiplying it by γ. (The convergence parameters α and γ are often chosen around 0.5.) (4) This concludes one epoch of self organization; repeat the operations a preestablished number of times (Often, seven to ten epochs of self organization are carried out.), or until a preestablished degree of convergence (as judged by changes in the vectors v_j) is attained. After the self organization is completed, we can evaluate the distance $d_{\max}$ between the two most distant vectors v_j, then evaluate the distance between every pair of vectors. When a pair of vectors is closer than some preestablished fraction of $d_{\max}$ (say one percent), then one of the pair can be eliminated without much loss in the capacity of the v_j to represent the input training exemplars. The remaining v_j define the centers $c_j, j = 0,...,N-1$ (where N may be smaller than its initially specified value).

The next step in training the MVLS network is to specify a width parameter β. Then based on a user-specified option, either identify the L nearest neighbors to each center vector c_j from among the set of input training exemplars, $x_j, z_j, j = 0,...,m_o - 1$, or identify all the nearest neighbors to the center vector from among the set of input training exemplars $x_j, z_j, j = 0,...,m_o - 1$, within a hypersphere of preestablished radius. Use these neighbors, and a weighted least squares approach to identify the coefficient vectors $x_j, z_j, j = 0,...,m_o - 1$. Each coefficient A_j corresponds to a center vector c_j; it identifies the linear model in the neighborhood of the center.

Let $x_i, z_i, i = 0,...,m - m_o - 1$, denote the exemplars that have been set aside for testing the MVLS network. Using each testing input x_i we can operate the MVLS network in feed forward mode to evaluate the predicted output, z_i. The prediction error is:

$$\varepsilon = \frac{1}{m - m_0} \sqrt{\sum_i \|z_i - y_i\|^2} \tag{13}$$

Given this error measure, there are many potential approaches to optimization of the MVLS network parameters. A simple approach, actually implemented in the software, is to allow the centers to self organize, then allow them to remain fixed. Vary the value of the width parameter β, and evaluate the linear term coefficients for each β. Optimize the network with respect to β. Then the network is optimal for the particular set of fixed centers. The analyst can repeat the process (randomizing the initial center locations at each iteration) till the error measure is satisfactorily minimized. A more complex approach would simultaneously vary the width parameter and the centers to optimize the MVLS network.

The MVLS network was implemented in MATLAB.

NUMERICAL EXAMPLE - APPLICATION TO A NONLINEAR HARDENING OSCILLATOR

To illustrate the effective utilization of the combined CVA-ANN approach, we use the nonlinear hardening oscillator of Figure 1 and Eq. (14), which is simulated using an analog computer.

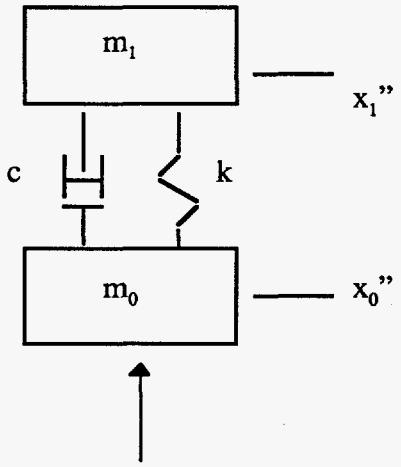

Figure 1. A nonlinear hardening oscillator.

$$\begin{aligned}
&\ddot{x}_1 + 2\zeta\omega_n(\dot{x}_1 - \dot{x}_0) + \omega_n^2(x_1 - x_0) + \alpha\omega_n^2(x_1 - x_0)^2 + \beta\omega_n^2(x_1 - x_0)|x_1 - x_0| = 0 \\
&\alpha = 3000 \\
&\beta = 3500 \\
&\omega_n = 2\pi(11.5) \\
&\zeta = 0.04
\end{aligned} \tag{14}$$

The terms c and k in Figure 1 indicate damping (dashpot) and stiffness (spring) behavior. Equation (14) defines the system dynamics in detail. The dashpot coefficient c in Figure 1 is $2\zeta\omega_n$ and the stiffness k at small values of $|x_1 - x_0|$ approaches ω_n^2, where the natural frequency is $f_n = 2\pi\omega_n = 11.5\,Hz$. At larger values of $|x_1 - x_0|$ the quadratic terms are dominant and the resonant frequency increases with increasing test level. For positive excursions of $x_1 - x_0$ the quadratic and absolute value terms add. For negative excursions they subtract. Since $\beta > \alpha$ we expect increasing stiffness in either direction, but less stiffness increase occurs in the direction of negative $x_1 - x_0$. In this experiment the level of the band limited random drive $\ddot{x}_0$ is adjusted to excite about equal root mean square (rms) responses in the linear and quadratic terms.

The response acceleration $\ddot{x}_1$ and the drive acceleration $\ddot{x}_0$ are digitized at 150 samples/second. The excitation and response were measured for a duration of about 14 seconds, so that about 2000 samples of the digitized excitation and response could be recorded. For the CVA analysis, past values of the time series are obtained by using six past values of the acceleration drive and response. A local CVA model with two states was used to compute the transformation map T.

The transition map relating a specific set of states to future states was modeled using an MVLS network. The first 1800 data points were set aside for training and testing of the MVLS network. The training set included 1600 input/output data points, and the testing set included 200 input/output data points. The input to the MVLS network was defined as the two system states at two consecutive times, and the input excitation at the current time and the

next consecutive time. That is, if we denote the states as $m_1(j), m_2(j), j = 0,...,1799$, and the input excitation as $\ddot{x}_0(j), j = 0,...,1799$, then an input to the ANN would be

$$(m_1(j-1), m_1(j), m_2(j-1), m_2(j), \ddot{x}_0(j), \ddot{x}_0(j+1)) \tag{15}$$

for j=1,...,1799. The corresponding ANN output would be

$$(m_1(j+1), m_2(j+1)) \tag{16}$$

An MVLS network with 30 centers was specified, and the 45 nearest neighbors were used to train each local linear model. The predictions of ANN outputs based on ANN inputs were made using the three local linear models with centers nearest to the input. A training error (Eq. (13)) of about 0.20 was achieved on the normalized data.

The MVLS network predictions are plotted versus the corresponding testing exemplars for the two states, in Figures 2a and 2b. If the MVLS network prediction were a perfect match to the exemplars, then all the data in the figures would fall along the diagonal. However, the ANN provides only an approximate representation of the physical system, therefore, the representation is imperfect. The representation could be improved, in principle, by increasing the number of centers, the number of exemplars, etc., but then training would take longer, the ANN would run longer, etc.

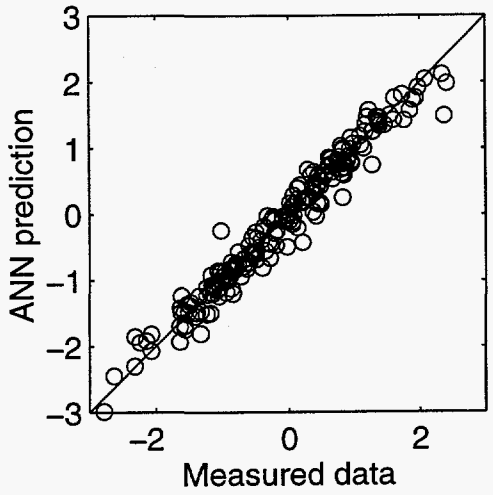

Figure 2a. MVLS network predictions versus training exemplars for state 1.

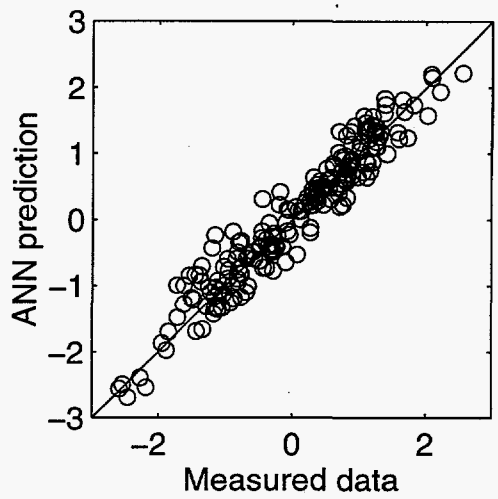

Figure 2b. MVLS network predictions versus training exemplars for state 2.

Following training, the MVLS network was used to predict the response of the nonlinear system in Figure 1. A response prediction was executed using some of the transformed data not previously used for training or testing. To perform the prediction, the MVLS network was provided with the initial conditions and the input excitation. The response was computed over the duration of the input, 256 points. The results are shown in Figure 3, for state 1. The solid line shows the transformed measured data, and the dashed line shows the MVLS network prediction. Though the prediction appears to stay in phase with the transformed measured signal, this is not always the case.

To further judge the quality of the prediction relative to the transformed measured data, some measures of the time domain responses were computed. Figure 4 shows the estimated spectral densities of the measurement-based and the predicted states. The former curve is shown as solid, and the latter is shown as dashed. The match is good, particularly where the mean square power is high. The simulation apparently has some high frequency harmonics at very low level where none exist in the measurement-based signal. Figure 5 shows the kernel density estimators

(estimators of the probability density functions) of the measurement-based and the predicted states. The former curve is shown as solid, and the latter is shown as dashed. The match is fairly good. The amplitudes in the prediction tend to underpredict the measurement-based state.

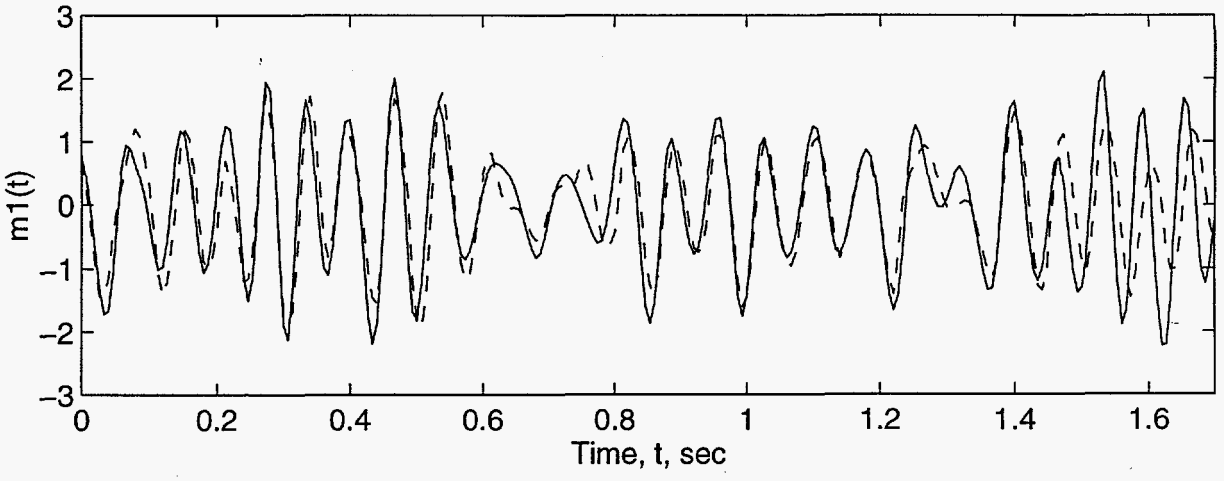

Figure 3. Time histories of state based on measured data (solid line) and predicted state (dashed line).

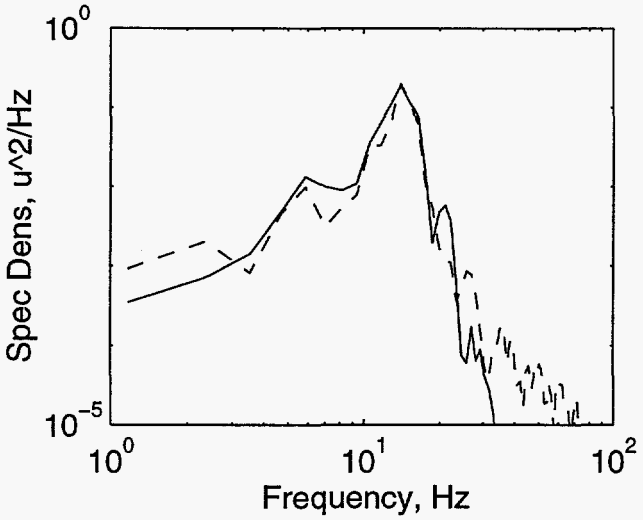

Figure 4. Estimated spectral densities of measurement-based state (solid line) and predicted state (dashed line).

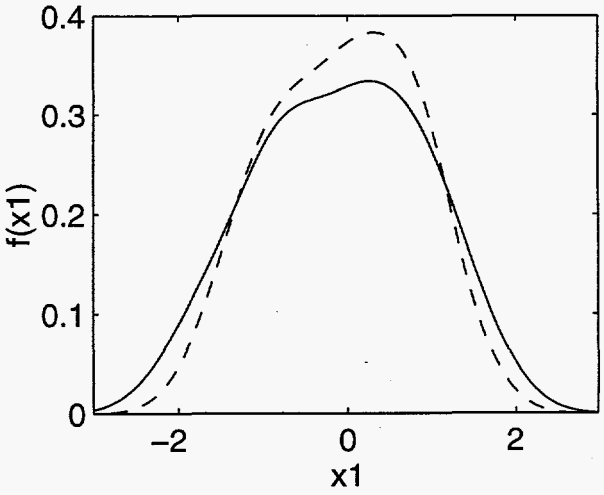

Figure 5. Kernel density estimators measurement-based state (solid line) and predicted state (dashed line).

CONCLUSIONS

A method for performing local linear analysis on a system to obtain nonlinear system response states, then modeling the temporal evolution of these states with an artificial neural network has been developed. Canonical variate analysis has been used to perform a local linear transformation of the response of a globally nonlinear system to obtain simple response states. The state evolution was then modeled using a multivariate linear spline network. An example shows that the method can yield accurate results.

Experience using canonical variate analysis in its traditional framework indicates that the use of an artificial neural network in the current predictive framework substantially speeds the prediction process. In fact, speed increases of approximately an order of magnitude , or more, have been realized.

Further, the use of an artificial neural network to predict states of system response rather than the response itself enables the practical use of artificial neural networks. The dimension of the network is reduced thereby diminishing the training data and training time requirements to the artificial neural network.

ACKNOWLEDGMENT

This work was sponsored by the United States Department of Energy under contracts with Sandia National Laboratories and Los Alamos National Laboratory.

REFERENCES

Farmer, J., (1988), "Exploiting Chaos to Predict the Future and Reduce Noise," in *Evolution, Learning and Cognition*, Y. C. Lee, Ed., World Scientific, Singapore, pp. 277-330.

Jones, R. D., et. al., (1990), "Nonlinear Adaptive Networks: A Little Theory, A Few Applications," Cognitive Modeling in System Control, The Santa Fe Institute.

Larimore, W., (1983), "System Identification, Reduced Order Filtering, and Modeling Via Canonical Variate Analysis," *Proceedings of the 1983 American Control Conference*, H. S. Rao and P. Dorato, Eds., 1982, pp. 445-451.

Larimore, W., (1991), "Identification and Filtering of Nonlinear Systems Using Canonical Variate Analysis," In *Nonlinear Prediction and Modeling*, M. Casdagli and S. Eubank, Eds., Addison Wesley. A Proceedings Volume in the Santa Fe Institute Studies in the Sciences in Complexity.

Moody, J., Darken, C., (1989), "Fast Learning in Networks of Locally-Tuned Processing Units," Neural Computation, V. 1, pp. 281-294.

Packard, N., Crutchfield, J., Farmer, J., Shaw, R., (1980), "Geometry from a Time Series," *Physical Review Letters*, V. 45, No. 9.

Paez, T., Tucker, S., O'Gorman, C., (1997a), "Complex Structure Simulation with Neural Networks," *Intelligent Civil Engineering Materials and Structures*, ASCE, New York.

Paez, T., O'Gorman, Tucker, S., (1997b), "Simulation of Nonlinear Random Vibrations Using Artificial Neural Networks," *Proceedings of the Sixth International Conference on Recent Advances in Structural Dynamics*, Institute of Sound and Vibration Research, Southampton, United Kingdom.

Priestly, M., (1980), "State Dependent Models: A General Approach to Nonlinear Time Series Analysis," *Journal of Time Series Analysis*, V. 1, pp. 47-71.

Tong, H., (1990), *Nonlinear Time Series Analysis - A Dynamical Systems Approach*, Clarendon Press, Oxford, 1990.