Table 1 Singular values of Berman's ΔM matrices for the second case (noisy data)

Number of measured modes m									
1	2	3	4	5	6	7	8	9	10
0.0219	**0.0390**	**0.0739**	**0.1001**	**0.1001**	**0.1001**	**0.1001**	**0.1001**	**0.1001**	**0.1001**
0.0000	**0.0085**	**0.0134**	**0.0133**	**0.0134**	**0.0250**	**0.0249**	**0.0250**	**0.0250**	**0.0250**
0.0000	0.0000	**0.0029**	**0.0030**	**0.0036**	**0.0058**	**0.0074**	**0.0100**	**0.0198**	**0.0199**
0.0000	0.0000	0.0000	**0.0024**	**0.0025**	**0.0032**	**0.0032**	**0.0036**	**0.0099**	**0.0101**
0.0000	0.0000	0.0000	0.0000	0.0002	0.0003	0.0005	0.0006	0.0007	0.0009
0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0002	0.0005	0.0007	0.0007
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0002	0.0006	0.0006
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0003	0.0003
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001	0.0001
0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0000	0.0001

man's ΔM matrices were determined by the singular value decomposition and are plotted in Fig. 2, where it can be seen that the rank of Berman's ΔM matrix was 3 when three or more measured modes were available.

The second case study was carried out on the same analytical model. In this case mass errors at points 2, 3, 5, and 8 and stiffness errors between points 2 and 3, 5 and 6, and 6 and 7 have been introduced. Berman's ΔM matrices were calculated using 1–10 modes for the actual system and their ranks were determined by the singular value decomposition. Again, the rank of Berman's ΔM matrix was equal to the number of mass error sites when sufficient (four or more) measured modes were available and was not affected by the number of stiffness errors made on the structure. Table 1 shows the singular values of Berman's ΔM matrix when the measured mode shape vectors were contaminated by 2% Gaussian random error. It can be seen that the numerical rank was 4 when four or more measured modes were available. A careful definition of numerical rank has been given in a paper by Golub et al.[5] The essential idea is briefly described as follows. We were going to look at the nonzero singular values of Berman's ΔM matrices and chose a number δ as a zero threshold. The choice of δ was based on measurement errors (information about the uncertainty of the measured data) incurred in estimating the coefficients of those Berman ΔM matrices. In this case 2% random error was applied to the measured mode shape vectors so that by considering the roundoff and the measurement errors in the matrix computation the value of δ was set to 0.02 times the spectral norm of Berman's ΔM matrix.

IV. Concluding Remarks

It is proved that in general Berman's ΔM matrix is a projection of the actual mass error matrix in an $n \times n$ subspace when sufficient modes are available. And so the number of mass error sites can be determined when Berman's ΔM matrix is factorized using the singular value decomposition. This technique would not be affected by the number of stiffness errors for any system. If the measured mode shape vectors are slightly contaminated by measurement noise, the numerical rank can be used as an indicator for the number of mass error sites. The noise introduced in the measurement of natural frequencies will not affect the determination of Berman's ΔM matrix and hence has no effect of its rank.

On the contrary, Baruch's ΔK matrix is able to determine the rank of an actual stiffness error matrix only when sufficient modes are available and the correct mass matrix is used. This is because Baruch[1] derived his formula using a known symmetric positive definite mass matrix, so inherently the correct mass matrix was used to obtain the Baruch's ΔK matrix. His formula for a stiffness error matrix is given as follows:

$$[\Delta K_{\text{Baruch}}] \triangleq -[K_A][\Phi][\Phi]^T[M_A] - [M_A][\Phi][\Phi]^T[K_A] + [M_A][\Phi][\Phi]^T[K_A][\Phi][\Phi]^T[M_A] + [M_A][\Phi][\Omega][\Phi]^T[K_A]$$

From the preceding equation, one can observe that if both sets of measured natural frequencies and mode shape vectors are contaminated by measurement noise, the rank of Baruch's ΔK matrix may be affected.

Acknowledgments

The author gratefully acknowledges the financial support of the Croucher Foundation in Hong Kong. He also wishes to thank D. J. Ewins at Imperial College for supervising this work.

References

[1]Baruch, M., and Itzhack, I. Y. B., "Optimal Weighted Orthogonalization of Measured Modes," *AIAA Journal*, Vol. 16, No. 4, 1978, pp. 346–351.

[2]Berman, A., "Mass Matrix Correction Using an Incomplete Set of Measured Modes," *AIAA Journal*, Vol. 17, No. 10, 1979, pp. 1147, 1148.

[3]Wei, F. S., "Stiffness Matrix Correction from Incomplete Test Data," *AIAA Journal*, Vol. 18, No. 10, 1980, pp. 1274, 1275.

[4]To, W. M., and Ewins, D. J., "A Criterion for the Localization of Structural Modification Sites Using Modal Data," *Proceedings of the 8th International Modal Analysis Conference*, Vol. 2, 1990, pp. 961–967.

[5]Golub, G. H., Klema, V. C., and Stewart, G. W., "Rank Degeneracy and Least Squares Problems," Dept. of Computer Science, Stanford Univ., Tech. Rept. STAN-CS-76-559, Stanford, CA, Aug. 1976.

Space Shuttle Main Engine Sensor Modeling Using Vector Quantization

Timothy F. Doniere* and Atam P. Dhawan†
University of Cincinnati, Cincinnati, Ohio 45221

Introduction

ARTIFICIAL neural networks have been used to model Space Shuttle main engine (SSME) sensor parameters.[1,2] A neural network sensor model predicts the value of the sensor parameter based on a selected set of other sensor measurements. The errors, or differences between the values predicted by the model and the actual sensor values, provide information on the health of the sensor.

Developing accurate models is particularly difficult due to the highly complex, nonlinear nature of SSME, the limited suite of measured parameters, and the large variability of behavior among engines of the same design. Neural networks are well suited for problems in which the exact relationships between inputs and outputs are complex or unknown, if the system state is sufficiently represented in the inputs.[3] Feedforward neural networks have been effectively used to model critical parameters of the SSME during the startup transient.[4]

Received Oct. 1, 1993; revision received April 14, 1994; accepted for publication April 15, 1994.

*Graduate Student, Electrical and Computer Engineering Department.

†Associate Professor, Electrical and Computer Engineering Department, Rhodes Hall, ML30.

In time series prediction using a function approximator model, a time window of each input parameter is typically used to provide time dependent information. The size of the window multiplies the number of inputs to the network. To insure good generalization by the model, data from a number of the SSME test firings must be utilized in training. This results in a large number of training vectors particularly when a time window for each input parameter is used. A large set of training vectors not only causes excessive training times, it may also result in nonconvergence of the network model. Therefore, it is desirable to obtain a reasonable number of training vectors which are derived from several SSME test firings.

In this Note we provide a data compression scheme using the learning vector quantization (LVQ) algorithm.[5] The design issues and methodology to model a SSME parameter, the high pressure oxidizer turbine discharge temperature, are presented. A feedforward neural network architecture with the Quickprop[6] training algorithm is used to develop the sensor model. The Quickprop training algorithm was derived from the standard backpropagation algorithm.[7] Its advantage over standard backpropagation is that it typically trains an order of magnitude faster. The performance of this sensor model trained with full data is compared with the performance of the sensor model trained with the data reduced using the LVQ algorithm. In addition, the LVQ paradigm is also investigated as a sensor model. The performance of LVQ model is compared with the feedforward neural network model.

Learning Vector Quantization

LVQ is an auto associative, supervised nearest-neighbor classifier. Using competitive learning, this paradigm can classify the input vector as one of the output classes by developing decision surfaces in the multidimensional feature space. It is a special case of the self-organizing feature map[5] which optimally assigns K reference vectors $\boldsymbol{m}_i$ such that the density of each $\boldsymbol{m}_i$ best represents the density function $p(x)$.

In general, the LVQ network can be implemented by a two-layer feedforward topology consisting of an input and an output layer. The output layer contains the codebook vectors which can be initialized using either K random vectors or arbitrary samples taken from the data set. Alternately, vectors chosen from the data set based on a priori knowledge can be used to initialize the codebook vectors. This procedure may provide faster convergence and better performance. In the learning process, the training vectors are iteratively presented to the input layer. When an input vector is presented it is compared with the codebook vectors. The error between the input vector and the nearest (Euclidean distance) codebook vector is computed. Based on the learning rate, a small part of this error is used to update the codebook vector such that the codebook vector more closely resembles the input vector. The iterative process of training is continued until an error criterion is reached.

The mathematical representation of the learning paradigm just described is outlined as follows:

$$\boldsymbol{m}_c(t+1) = \boldsymbol{m}_c(t) + \alpha(t)\,[\boldsymbol{x}(t) - \boldsymbol{m}_c(t)]$$

where

$$c = \arg\min_i \|\boldsymbol{x} - \boldsymbol{m}_i\|$$

and $0 < \alpha(t) < 1$. The learning rate $\alpha(t)$ is linearly decreased as the number of iterations increases.

Application to Space Shuttle Main Engine Data

In the LVQ model of the SSME temperature parameter, assume that there are n SSME parameters to be used as inputs. Each vector will contain $(n+1)$ elements, where the $(n+1)$th element is the label for that vector. The label represents the value of the modeled parameter. The input layer of the LVQ network will contain $(n+1)$ nodes. The output layer will consist of K nodes depending on the ratio of data compression. The data used in the models are normalized to insure that all inputs have an equal chance of affecting the output.

The result of training the LVQ network is a set of vectors called the codebook vectors. The density of the set of codebook vectors represents the density of the set of training vectors. The codebook vectors can then be used as a training set for other neural network paradigms or directly used to estimate the desired parameter.

The LVQ estimator scheme uses the codebook vectors to estimate the desired parameter. Each input pattern is presented to the network, and a neighborhood of the closest codebook vectors is found. The estimate is computed from a nonlinear sum of each codebook vector label. The weight of each label is a function of the distance of the codebook vector from the input vector. This scheme was described by Szewczyk and Hajela.[8]

Results and Discussion

Four SSME ground test firings were used for training, for a total of 560 training vectors. Seven SSME test firings were used as validation sets. A time window of five past values was used for each input variable. To evaluate the performance of the compressed training data, feedforward neural networks were trained using the Quickprop training algorithm. Each network had one hidden layer with 10 processing units. The training data was compressed to 50% and 20% of the original data. This is accomplished by using 280 and 112 codebook vectors, respectively. The codebook vectors were initialized using samples from the training set. For each compression ratio, the neural network model for the SSME temperature parameter was evaluated using the seven validation sets.

Table 1 Error statistics for the training and validation sets for the Quickprop trained with full data, 50%, and 20% of the data

	Full		50%		20%	
Test No.	Nrms	Max%	Nrms	Max%	Nrms	Max%
1060	0.021	7.531	0.028	13.052	0.029	14.015
1066	0.018	4.284	0.024	7.564	0.023	7.410
1070	0.023	6.006	0.024	10.257	0.024	9.942
1077	0.014	5.532	0.014	7.957	0.015	7.955
1061	0.013	7.352	0.021	13.896	0.023	15.029
1062	0.022	6.872	0.026	11.410	0.027	12.405
1063	0.024	6.060	0.029	9.189	0.028	8.909
1067	0.024	5.180	0.029	7.242	0.029	7.516
1071	0.042	14.958	0.050	16.575	0.050	16.102
1072	0.027	4.946	0.027	9.162	0.028	9.367
1075	0.021	8.799	0.027	14.198	0.029	14.059

Table 1 Error statistics for the training and validation sets for the LVQ estimator trained with full data, 50%, and 20% of the data

	Full		50%		20%	
Test No.	Nrms	Max%	Nrms	Max%	Nrms	Max%
1060	0.001	0.147	0.013	7.191	0.019	8.392
1066	0.000	0.067	0.007	4.132	0.014	4.417
1070	0.000	0.088	0.009	4.897	0.017	5.653
1077	0.000	0.102	0.009	5.538	0.011	4.891
1061	0.017	10.950	0.015	8.834	0.015	9.540
1062	0.014	5.450	0.015	5.872	0.018	8.276
1063	0.022	8.104	0.023	7.310	0.023	7.621
1067	0.019	7.626	0.020	6.604	0.021	6.195
1071	0.027	12.241	0.029	15.749	0.035	15.993
1072	0.017	7.735	0.017	7.201	0.022	6.217
1075	0.017	10.896	0.020	10.903	0.022	11.025

Table 3 Training time, CPU s

Compression ratio	LVQ	Backpropagation	Backpropagation +LVQ
1:1	—	519	519
2:1	135	263	398
5:1	57	107	164

The errors associated with the network models are represented by the normalized root mean square (nrms) error and the maximum percent error. The nrms error is the typical rms normalized using the sum of squares of the desired outputs. The error statistics reported represent the average statistics for 10 networks trained with different weight initializations. Table 1 contains the error statistics for Quickprop trained with the full training set and the compressed training sets. Table 2 contains the error statistics for the LVQ estimator using full and compressed training sets. The first four rows in each table are for the training data.

Training times are shown in Table 3. The first two columns show the training times for LVQ and Quickprop. The last column indicates the total training time, that is, the time to reduce the training set added to the time for Quickprop training using the reduced training sets.

Conclusion

In this investigation, the issue of reducing the training time of a feedforward neural network with backpropagation was addressed. To improve training efficiency, the networks were trained on a reduced data set. The data reduction was obtained using the LVQ algorithm. Various compression ratios were investigated to compress the training data. The impact of various compression ratios on training was analyzed through the comparison of error statistics on the validation data.

It was demonstrated that the compressed data set could be used to train other networks or as the codebook set for the LVQ estimator. It should be noted that the LVQ, when used as a neural network model, can provide much faster estimation. However, the output of the LVQ estimator was noisy with larger errors. These errors are primarily because of artifacts introduced by the formation of the codebook vectors. If an acceptable interval of errors in estimation is predefined, the noise in the output of the LVQ estimator can be controlled by designing an appropriate number of codebook vectors. Thus, if the estimation time is of primary concern, the LVQ neural network model could be used as an estimator.

It can be concluded from the presented investigation that the training efficiency of the feedforward neural network model can be improved by reducing the training data using the LVQ algorithm. As training data was reduced, there was a comparatively small increase in the errors, whereas the training times were decreased significantly. It should be noted that such an improvement in efficiency by reducing the training time may be very useful for very large data sets, such as the data compiled during the SSME mainstage operation.

Acknowledgments

This work was partially supported by grants from the NASA Lewis Research Center and NASA-UC Space Engineering Research Center for System Health Management Technology at the University of Cincinnati. The authors gratefully acknowledge the help and valuable contributions of Claudia Meyer of Sverdrup Technology Inc. The Quickprop training algorithm is the C++ version developed by Charles Peck. It is a modified version of Terry Regier's C implementation of Scott Fahlman's Quickprop training algorithm.

References

[1]Bickmore, T., "Probabilistic Approach to Sensor Data Validation," AIAA Paper 92-3163, July 1992.

[2]Makel, D. K., Flaspohler, W. H., and Bickmore, T. W., "Sensor Data Validation and Reconstruction, Phase 1: System Architecture Study," NASA CR 187122, 1991.

[3]Chen, S., Billings, S. A., and Grant, P. M., "Non-linear Systems Identification Using Neural Networks," Univ. of Edinburgh, Research Rept. 370, Edinburgh, Scotland, UK, Aug. 1989.

[4]Meyer, C. M., and Maul, W. A., "The Application of Neural Networks to the SSME Startup Transient," AIAA Paper 91-2530, June 1991.

[5]Kohonen, T., *Self-Organization and Associative Memory*, Springer-Verlag, Berlin, 1989, pp. 119–157, 199–202.

[6]Fahlman, S. E., "An Empirical Study of Learning Speed in Back-Propagation Networks," Carnegie Mellon Univ., Technical Rept. CMU-CS-88-162, Pittsburgh, PA, Sept. 1988.

[7]Rumelhart D. E., Hinton, G. E., and Williams, R. J., "Learning Internal Representations by Error Propagation," *Parallel Distributed Processing Explorations in the Microstructure of Cognition*, edited by D. E. Rumelhart, J. L. McClelland, and the PDP Research Group, Vol. 1: Foundations, MIT Press, Cambridge, MA, 1986, pp. 318–364.

[8]Szewczyk, Z., and Hajela, P., "Neurocomputing Strategies In Decomposition Structural Design," *AIAA/ASME/ASCE/AHS/ASC 34th Structures, Structural Dynamics, and Materials Conference, AIAA/ASME Adaptive Structures Forum*, Vol. 4, AIAA, Washington, DC, 1993, pp. 2458–2465.

Simplified Analysis of General Instability of Stiffened Shells with Cutouts in Pure Bending

Dimitris L. Karabalis*
University of South Carolina, Columbia, South Carolina 29208

Introduction

FAILURE of stiffened shells in a general instability mode occurs when the spring constant of the transverse stiffening frames is reduced below a minimum necessary value for the confinement of instability failure between adjacent frames. Under such conditions, general instability failure can be spread over a number of frame spacings in a waveform shape described as an "inward bulge" occurring at the extreme compression fibers. A comprehensive discussion of the complex problem of general instability of stiffened shells in bending, pressure, torsion, transverse shear, and combined bending and torsion has been reported by Becker.[1] His formulation of the problem, based on a correlation of analytical and experimental results obtained prior to 1958, has become the backbone of most of industry's standard design manuals, e.g., Bruhn.[2] Furthermore, a number of analytical and experimental studies on the subject of general instability of stiffened or unstiffened shells with or without cutouts are available.[3,4] For design purposes, Shanley[5] has proposed a simple criterion which is based on a correlation of analytical results obtained from a simplified model and experimental data. Similar criteria have also been proposed by Gerard[6] and Hoff.[7] However, one should be aware of the fact that all of these general instability criteria are based on tests which may not be representative of contemporary fuselage designs.[8,9] Despite the criticism, Shanley's criterion remains in general use due to its simplicity, its direct correlation to experimental data, and the relatively few and easily obtainable factors involved in the calculations. However, one distinct limitation of the aforementioned criteria is that they are not directly applicable to incomplete frames at locations where substantial fuselage cutouts, such as cargo doors, are present. In this work, a simple analytical procedure is presented which may be useful in checking the design of the stiffening frames of cylindrical fuselages, with or without cutouts, for failure by general instability under bending conditions. The proposed formulation traces the basic steps of Shanley's original work for complete circular frames and extends it to incomplete frames, with or without edge stiffeners at the perimeter of the cutout.

Formulation

Simplified Model

The basic model used in Ref. 5 for general instability of a fuselage structure is a spring-bar assembly such as the one shown in Fig. 1. The horizontal bars represent the sheet-stringer elements

Received June 14, 1993; revision received Nov. 4, 1993; accepted for publication Nov. 16, 1993.

*Associate Professor, Department of Civil Engineering, and Consultant, Gulfstream Aerospace Corporation, P. O. Box 2206, Savannah, GA 31402.