Pergamon

Computers ind. Engng Vol. 30, No. 2, pp. 227-242, 1996
Copyright © 1996 Published by Elsevier Science Ltd
Printed in Great Britain. All rights reserved
0360-8352/96 $15.00 + 0.00

0360-8352(95)00168-9

PERFORMANCE ANALYSIS OF ROTATING MACHINERY USING ENHANCED CEREBELLAR MODEL ARTICULATION CONTROLLER (E-CMAC) NEURAL NETWORKS

CHANG-CHING (DAVID) LIN and HSU-PIN (BEN) WANG

Department of Industrial Engineering, FAMU/FSU College of Engineering, 2525 Pottsdamer St, Tallahassee, FL 32310-6046, U.S.A.

Abstract—For today's sophisticated machinery systems, on-line predictive maintenance has become a most reliable and cost effective method for machinery maintenance. An effective fault monitoring and diagnosis tool is able to recognize the characteristics, conditions, and developing trends of an operating machinery system, and is able to estimate fault severity of the system quantitatively. In order to meet the requirements, a new approach combining cerebellar model articulation controller (CMAC) neural networks and advanced vibration monitoring methods has been presented. A test rig consisting of two rotating hubs driven by a d.c. motor was used to produce different machine imbalance levels. A two-stage experiment is proposed. First, a CMAC network is trained with pre-defined imbalance conditions and a test data set is given to check the trained CMAC for imbalance severity prediction. Second, a CMAC network is trained by using a set of different imbalance levels and is then given an untrained condition to test the CMACs ability of imbalance severity interpolation. In addition, several potential capabilities of CMAC networks will be discussed and shown. The properties of the CMAC are notable. If implemented properly, CMAC can be used as an effective machinery condition monitor and fault severity estimation tool.

1. INTRODUCTION

Predictive maintenance is the science of using various kinds of data to determine the condition of machinery and to predict failure before it occurs. The benefits of predictive maintenance are reduced downtime, lower maintenance costs and improved safety. For today's sophisticated systems of machinery, predictive maintenance has become a most reliable and cost-effective method for monitoring and diagnosing faults [1]. Predictive maintenance relies on instruments with specific sensors to monitor a machine's normal and abnormal operations and to analyze these signals by comparing them with baseline data. There are many different types of methods that may be used, including oil analysis, vibration analysis and temperature and pressure monitoring. For many years, vibration analysis has been widely accepted as the most reliable method for predicting machinery problems [2, 3]. In this paper, vibration signals are used for rotating machine condition monitoring, fault diagnosis and severity estimation.

Trend monitoring and fault recognition are two major data anlysis techniques required for most condition monitoring and diagnostic systems [1]. Trend monitoring involves plotting functional parameters (e.g. vibration, cutting force, current, temperature, pressure etc.) or their trending indices (e.g. RMS, Kurtosis etc.) against operating time. Fault recognition normally requires a detailed analysis of machinery signals to identify specific fault patterns. Traditionally, this is performed through visual inspection by experienced personnel using spectrum analysis or associated signal processing methods. However, these methods are usually costly and very inefficient. As an alternative to conventional fault diagnostic methods, artificial intelligence techniques are being introduced to assist in fault diagnosis. Several expert systems [4, 5] were developed to diagnose the malfunctions of advanced production equipment. Clearly, a rule-based expert system requires a complex database that needs to be generated and maintained by experts familiar with the historical causes and patterns of machine failures [6]. Therefore, the application of expert systems in manufacturing systems is restricted.

An effective fault diagnostic tool must be able to recognize the characteristics, conditions, and developing trends of an operating machinery system and be able to predict its future conditions. It must be able to function under uncertain environments. To meet such requirements, artificial neural networks (ANN) have a number of potential advantages. In recent years, great interest has been generated concerning the applications of ANNs. Researchers reported that ANNs worked well for many real-world classification problems [7]. Knapp and Wang developed a neural network using the backpropagation (BP) learning algorithm to classify the faults of a machine based on vibration signals [8]. Kim *et al.* conducted a number of multiple machinery fault experiments using BP neural networks for location detection, and severity determination based on vibration frequency spectra [9]. Lin and Wang studied a methodology which employed the autoregressive (AR) parametric modeling technique for vibration signals accompanying the adaptive resonance theory 2 (ART2) neural network to perform signal classification [10]. Liu and Iyer used a backpropagation neural network to diagnose roller bearing defects [11]. However, BP has two major disadvantages with real-time applications; a low converging speed and an inability of incremental learning, mainly due to its global generalization property [12]. While the ART2 neural network looks promising for incremental and faster learning [10], its drawback is the ART2 algorithm which is only a pattern classifier, making machine fault severity estimation and fault prediction impossible. For most real-world problems, quantitative description of machinery behavior is important and often essential.

In this paper, a new approach to machine fault severity estimation has been developed. A test rig consisting of two rotating hubs driven by a d.c. motor was used to produce different machine imbalance levels. Vibration signals were acquired and then analyzed based on vibration trending indices and autoregressive (AR) model parameters. A CMAC neural network was employed to evaluate different machine imbalance levels quantitatively. This has been accomplished mainly due to the outstanding function approximation abilities and the incremental, speedy learning of the CMAC. For vibration monitoring, several machine trending indices were used as inputs to the CMAC. Alternatively, AR parameters were also used as inputs to the CMAC. A comparison result from these two experiments will be given. Finally, the properties of the CMAC neural networks will be investigated and studied.

2. CEREBELLAR MODEL ARTICULATION CONTROLLER (CMAC)

2.1. Literature review and background

The cerebellar model articulation controller or cerebellar model arithmetic computer (CMAC) was first proposed by Albus [13, 14]. Originally, the mathematical model of the CMAC was developed by imitating the information processing characteristics of the human cerebellum [15]. It has often been overlooked and traditionally regarded as a perceptron-like neural network that performed a table look-up of a nonlinear function. Actually, the CMAC is capable of very fast learning and contains certain features of interpolation and approximation [16]. It can learn nonlinear relationships from a very broad category of functions and generally converges in a small number of iterations. Several researchers have applied the CMAC to find solutions to various problems. For example, Moody used a modified multi-resolution hierarchy CMAC to predict chaotic time series [17]. Miller applied the CMAC to real-time control of a vision-assisted robot [18, 19]. Carter and others studied the fault tolerance of CMAC networks [20]. Linse and Stengel compared the function approximation capability of the CMAC with the backpropagation and the B-spline interpolation procedure [21]. Lee and Kramer developed a pattern discrimination model (PDM) based on the CMAC networks to analyze robot system degradation [22]. It was also proved by Wong and Sideris that the CMAC learning algorithms always converge with exponential speed in certain conditions [16].

2.2. Mathematical model of the CMAC

2.2.1. Nomenclature.

N = dimensions of input vector,

R = possible different values on each input dimension (input space resolution),

R^N = possible number of input vectors,

K = number of quantizing functions in each input dimension,
Q = resolution number for each quantizing function,
s_i = the ith input variable,
$\mathbf{S}$ = input vector (e.g. $\mathbf{S} = \{s_1, s_2, \ldots, s_N\}$),
$\mathbf{m}_i$ = the ith quantizing function,
$\mathbf{M}$ = a set of quantizing functions (e.g. $\mathbf{M} = \{\mathbf{m}_1, \mathbf{m}_2, \ldots, \mathbf{m}_N\}$),
$\mathbf{A}$ = a set of memory association tables or granule cells,
$\mathbf{A}_\mathrm{P}$ = a set of physical memory association tables,
$|\mathbf{A}_\mathrm{P}|$ = size of physical memory association tables,
p = CMAC output,
$\mathbf{P}$ = CMAC output vector,
p_d = desired output,
$\mathbf{P}_\mathrm{d}$ = desired output vector,
$\mathbf{k}_{ij}$ = a quantizing function consisting of a set of mossy fibers,
$\mathbf{g}_j$ = the jth set of granule cells,
ϵ = acceptable error,
β = percentage of correction,
w_{jo} = the old value of weight at address location j,
w_{jn} = the new value of weight at address location j,

where $i = 1, 2, \ldots, N, j = 1, 2, \ldots, K$.

2.2.2. CMAC mapping scheme. Based on the knowledge of the structures and functions of the various cells and fiber types in the cerebellum, Albus proposed a mathematical model to explain the information-processing procedure of the cerebellum. Basically, the CMAC is defined by a series of mappings:

$$\mathbf{S} \rightarrow \mathbf{M} \rightarrow \mathbf{A} \rightarrow \mathbf{A}_\mathrm{P} \rightarrow p, \qquad (1)$$

where $\mathbf{S}$ is an input vector, $\mathbf{M}$ is a set of codes (named mossy fibers in the CMAC) used to encode $\mathbf{S}$, $\mathbf{A}$ is a set of cells (called granule cells in the CMAC) contacted by $\mathbf{M}$, $\mathbf{A}_\mathrm{P}$ is a set of physical memory tables, and p is an output value. In CMAC mathematical formalism, $\mathbf{M}$ represents a set of quantizing functions and $\mathbf{A}$ is a set of memory association tables or address arrays. In the following, an example of a two-variable CMAC with four quantizing functions on each variable is provided to help illustrate the CMAC mapping algorithm. The example is shown in Fig. 1.

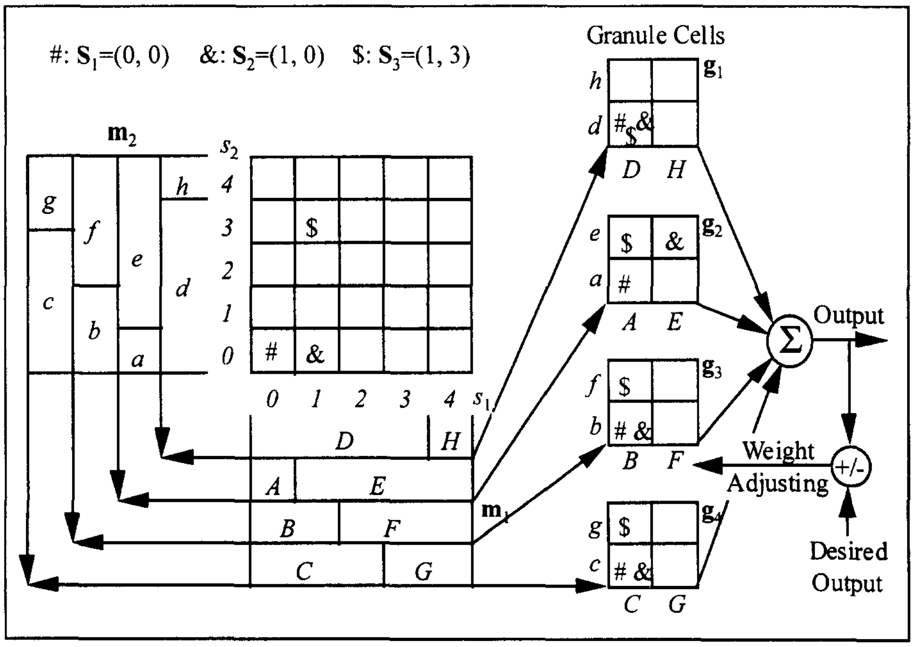

Fig. 1. Mapping scheme of a two-variable CMAC with four quantizing functions.

2.2.3. Input space discretization. An input vector **S** is the collection of N variables. In a real-world problem each variable may be associated with a measurement parameter, such as a reading from a sensor. Following the example in Fig. 1, a two-dimensional input space is represented by $\mathbf{S} = \{s_1, s_2\}$. For each dimension, a resolution number R is specified, meaning each input dimension is divided into R equal segments ($R = 5$ in the example). Depending on the different applications, the input space discretization level R may be high (fine resolution) or low (coarse resolution).

2.2.4. ***S** to **M** mapping.* The first CMAC mapping intends to simulate the function of mossy fibers in the cerebellum. Each individual input dimension has a unique mapping. Since N is the number of input dimension, therefore, there are N mappings as shown below:

$$\mathbf{S} \rightarrow \mathbf{M} = \begin{cases} s_1 \rightarrow \mathbf{m}_1, \\ s_2 \rightarrow \mathbf{m}_2, \\ \vdots \\ s_N \rightarrow \mathbf{m}_N. \end{cases} \tag{2}$$

In the CMAC, each s_i to $\mathbf{m}_i$ mapping may be defined by a set of K quantizing functions:

$$s_i \rightarrow \mathbf{m}_i = \{\mathbf{k}_{i1}, \mathbf{k}_{i2}, \ldots, \mathbf{k}_{iK}\}. \tag{3}$$

For the example in Fig. 1, the number of quantizing functions is four, so the s_1 to $\mathbf{m}_1$ mapping may be defined by four quantizing functions, $\mathbf{k}_{11}$–$\mathbf{k}_{14}$. Each quantizing function includes Q mossy fibers. Q is calculated by the following equations:

$$\begin{aligned} R &= 5, \\ K &= 4, \\ Q &= \mathrm{ceil}\left(\frac{R}{K}\right) = 2, \\ s_1 \rightarrow \mathbf{m}_1 &= \{\mathbf{k}_{11}, \mathbf{k}_{12}, \mathbf{k}_{13}, \mathbf{k}_{14}\}, \\ \mathbf{k}_{11} &= \{D, H\}, \\ \mathbf{k}_{12} &= \{A, E\}, \\ \mathbf{k}_{13} &= \{B, F\}, \\ \mathbf{k}_{14} &= \{C, G\}, \end{aligned} \tag{4}$$

where "A", "B", "C", "D", "E", "F", "G" and "H" label all mossy fibers in $\mathbf{m}_1$ and "ceil(R/K)" calculates the smallest integer larger than R/K, which is two in this example. Therefore, once a unique set of mossy fibers is given to each s_i to $\mathbf{m}_i$ mapping, the **S** to **M** mapping is completed. The number of the quantizing function K, also called the neighborhood number, is the most important parameter to be determined at this stage. K, which is arbitrarily chosen by the user, determines the generalization property of the CMAC.

2.2.5. ***M** to **A** mapping.* The second mapping, **M** to **A**, imitates the structure of the granule cells in the cerebellum. The granule cells are, in fact, a set of memory tables in the CMAC. As shown in Fig. 1, each level of quantizing functions from the two input dimensions forms a set of granule cells. Therefore, there is a total of four sets of granule cells; $\mathbf{g}_1, \mathbf{g}_2, \mathbf{g}_3, \mathbf{g}_4$. Note that the granule cell sets can be a hyper-cube for a higher input dimension case (e.g. $N > 3$). A granule cell is a weight storing location, or a pointer, pointing to a particular weight value in a practical computation implementation. Each granule cell receives inputs from different combinations of mossy fibers, so that each granule cell may be triggered by only one such combination. For example, three input vectors $\mathbf{S}_1 = (0, 0)$, $\mathbf{S}_2 = (1, 0)$ and $\mathbf{S}_3 = (1, 3)$ denoted by symbols " # ", "&" and "\$" respectively, are given in Fig. 1. $\mathbf{S}_1$ triggers the "D", "A", "B" and "C" mossy fibers in the $\mathbf{m}_1$ set quantizing functions, and "d", "a", "b" and "c" in the $\mathbf{m}_2$ set. By concatenating the same levels of mossy fibers, the "Dd", "Aa", "Bb" and "Cc" granule cells are chosen to store the weights of the $\mathbf{S}_1$ input vector. The output of the $\mathbf{S}_1$ vector is the summation of these distributed weights in granule cells, which concludes the **A** to p mapping. Following the same procedure, the weight of the $\mathbf{S}_2$ vector can be distributed into the "Dd", "Ee", "Bb" and "Cc" cells and $\mathbf{S}_3$ into the "Dd", "*Ae*", "*Bf*" and "*Cg*" cells. As shown in Fig. 1, S_1 and S_2 share three granule cells, while S_1 and S_3 share only one cell, because S_2 is closer to S_1 than S_3 is (see the input space in Fig. 1).

The memory size used in CMAC is KQ^N. In Fig. 1, KQ^N is 16, which is the total number of granule cells. Comparing KQ^N to actual memory size R^N (25 in the example), the CMAC uses less memory, and memory saving becomes significant when N is large. In addition, another memory reduction scheme was introduced by Albus, who applied a hash-coding technique to the CMAC networks [14]. Hash-coding is a commonly used computational method for reducing the amount of memory required to store sparse data space where a relatively small amount of data is scattered over a large number of memory locations. It operates by taking the memory address as a pseudo-random number. The number is usually restricted to the physical memory size. Hence, the last mapping in the CMAC, $\mathbf{A}$ to $\mathbf{A}_P$, is a many-to-few mapping. In reality, the size of $\mathbf{A}_P$ may be chosen to fill-up the available memory in the computer. However, the many-to-few property may lead to "data collision" problems. For the CMAC, this problem may be ignored because the effect is essentially identical to the already existing problem of learning interference in the CMAC, which is handled by iterative data learning [14].

2.2.6. Learning algorithm. The CMAC neural network is a supervised learning network. Thus, in the training stage, the desired output of each input vector must be given. The training consists of adjusting the values in the weight table based on the error between the present CMAC output and the desired output. Therefore, if $|p - p_d| > \epsilon$, where p is the current summation output, p_d is the desired output and ϵ is an acceptable error, then the learning begins. For each input–output pair, the weight adaptation process is described in equation (5):

$$w_{jn} = w_{jo} + \left(\frac{p - p_d}{K}\right) \times \beta. \tag{5}$$

In the equation above, w_{jn} is the new value of the weight at address location j, w_{jo} is the old value of the weight at address location j, β is the learning rate or the percentage of correction, and $j = 1, 2, \ldots, K$. The weight updating process terminates until $|p - p_d|$ is no larger than ϵ.

A procedure for learning a function in the CMAC is as follows:

1. Assume F is the function CMAC is to learn. Then $\mathbf{P}_d = F(\mathrm{S})$ is the desired input–output relationship.
2. Select n points from the input space where $\mathbf{P}_d$ is available.
3. Calculate the current CMAC output values by entering these n points, $\mathbf{P} = CMAC(\mathrm{S})$.
4. For every element in $\mathbf{P} = (p_1, p_2, \ldots, p_n)$ and in $\mathbf{P}_d = (p_{1d}, p_{2d}, \ldots, p_{nd})$, if $|p_i - p_{id}| < \epsilon$, then stop; the desired vaues have been stored. Otherwise, if any $|p_i - p_{id}| > \epsilon$, then adjust weights in the CMAC network using equation (5).
5. Go to Step 3 and continue updating weights until the CMAC output values of all n points selected meet the criterion of $|p_i - p_{id}| < \epsilon$.

3. AUTOREGRESSIVE (AR) PARAMETRIC MODEL

Traditionally, the Fast Fourier transform (FFT)-based spectral estimators are used to estimate the power spectral density (PSD) of time signals. Since Burg introduced the first parametric spectral estimation method [33], many parameter estimation methods have been developed [23]. Among them, the autoregressive (AR) modeling method is the most popular [24]. The major advantage of using the parametric spectral estimation method is its ability to translate a time signal into both frequency (PSD) domain and parameter domain [10]. In addition, parametric spectral estimation is based on a more realistic assumption about time series and does not need a long data collection to get a high-resolution spectrum.

An AR process is characterized by its AR parameters $\{a_1, a_2, a_3, \ldots, a_P, \sigma^2\}$. From the linear prediction viewpoint, if a random process is an AR_p (a p-order AR process), the unobserved sample of the random process x_n may be predicted based on the observed data set $\{x_{n-1}, x_{n-2}, x_{n-3}, \ldots, x_{n-p}\}$ (i.e. the previous p samples). Now consider a time series x_n:

$$x_n, n = -\infty, \ldots, 0, \ldots, \infty, \tag{6}$$

where the observed interval is from $n = 1, \ldots, N$. The autoregressive model of x_n can be defined as follows:

$$x_n = -a_1 x_{n-1} - a_2 x_{n-2} - \cdots - a_p x_{n-p} + e_n. \tag{7}$$

Here e_n is the prediction error, and p is the order of the model. Then, the parametric spectrum can be computed by plugging all $p\ a_k$ parameters into the theoretical power spectral density function defined in equation (8).

$$P_{\mathrm{AR}}(f) = \frac{2\Delta t \sigma^2}{\left|1 + \sum_{k=1}^{p} a_k \exp(-i2\pi f k \Delta t)\right|^2}$$
$$-\tfrac{1}{2} \leqslant f \leqslant \tfrac{1}{2}, \tag{8}$$
$$\Delta t = \frac{1}{S}.$$

Above, S is the sampling rate used in data acquisition, f is the fraction of the sampling rate, p is the prediction lag or order of the AR model, and σ^2 is the variance. If the prediction parameters a_k can be obtained, then the parametric spectrum $P_{\mathrm{AR}}(f)$ of the random process can be calculated through equation (8).

Several approaches are available for estimating the AR parameters. It has been observed that if the data consist of sinusoids with white noise, the peak location in the AR spectral estimate critically depends on the phase of the sinusoid [25]. The degree of phase dependence varies with different AR estimation methods. Of all the AR parameter estimation methods, the modified covariance method appears to yield the best estimation [26]. This is because it provides estimates which are the least sensitive to sinusoidal phases, and spectral peak shifting affected by noise is less than that of other methods [26]. An efficient algorithm for solving the equations of modified covariance methods was derived by Marple in 1980 [27]. Thus, after estimated parameters $\hat{a}_k$, for $k = 1, 2, \ldots, p$, are derived, the estimate of white noise variance is obtained. By plugging all these values (i.e. $\hat{a}_1, \ldots, \hat{a}_p, \hat{\sigma}^2$) into equation (8), the parametric spectrum of the random process may be obtained.

4. VIBRATION TRENDING INDEX TECHNIQUES

Vibration trending indices are good ways to translate vibration signals into measures of machinery health status. Each signal could have more than one trending index associated with it. Furthermore, each trending index, which may be treated as a different aspect of the signal, carries different sensitivities for different machine fault types. Several trending techniques have been developed and studied [3, 28–30]. In this paper, EWMA (exponential weighted moving average), RMS (root mean square), MFRMS (matched filter root mean square) and Kurtosis were chosen to monitor the vibration conditions of a rotating machine.

4.1. Exponentially weighted moving average (EWMA)

The EWMA method is used to monitor the condition of machines and determine when an abnormal condition is occurring. The EWMA statistic is a way of determining how much the observed signal differs from the signal of a machine operating under normal conditions, through the calculation of overall variance. Based on its AR model, the EWMA control statistic of a particular signal is defined as follows:

$$\mathrm{EWMA}_t = \max\{(1-\lambda)\mathrm{EWMA}_{t-1} + \lambda \ln[\rho^{\mathrm{fb}}_{\mathrm{normalized}}], 0\}, \tag{9}$$

where $\mathrm{EWMA}_0 = 0$, $\mathrm{EWMA}_t =$ the predicted EWMA value at time t (new EWMA) and $\mathrm{EWMA}_{t-1} =$ the predicted EWMA value at time $t-1$ (old EWMA); $\ln[\rho^{\mathrm{fb}}_{\mathrm{normalized}}]$ is the sample variance of observed values at time t, and λ is a smoothing constant, satisfying $0 < \lambda < 1$. The sample variance is calculated by forward and backward prediction errors of the AR model in the modified covariance method. The constant λ determines the ratio of the "memory" of the EWMA statistic. That is, λ determines the rate of decay of the weights, and in turn, the amount of information recollected from past data. It was determined experimentally in the research that the value of λ that minimized the predictive error was 0.7.

4.2. *Root mean square (RMS)*

The RMS method is the simplest approach for the measurement of overall vibration level. It is reported to be a useful performance measuring tool for machine components [31]. The root mean square value of a process can be calculated as in equation (10):

$$\mathrm{RMS} = \sqrt{\frac{1}{N}\sum_{n=1}^{N}(x_n - \bar{x})^2}, \tag{10}$$

where x_n is a stationary random process, the observed time interval $n = 1, \ldots, N$, and $\bar{x}$ is the mean of the random process.

4.3. *Kurtosis*

A localized defect on the surface of a rolling element produces a series of spikes on vibration signals. These spikes are usually superimposed onto the background vibration of the rolling element. This phenomenon calls for the application of statistical analysis, such as Kurtosis, to monitor the rolling machine. Kurtosis, the normalized fourth moment distribution statistic, has been found to be a good indicator of examining the acceleration distribution of a rotating element [28]. The equation of Kurtosis is defined below:

$$\mathrm{Kurtosis} = \frac{1}{N}\sum_{n=1}^{N}\frac{(x_n - \bar{x})^4}{\sigma^4}, \tag{11}$$

where σ is the standard deviation of the x_n series and $\bar{x}$ is the mean of the x_n series.

4.4. *Matched filter root mean square (MFRMS)*

The MFRMS trending index, which is calculated based on the AR parametric spectrum, has proved to be a good indicator of bearing damage [32]. The MFRMS is mathematically described in equation (12):

$$\begin{gathered}\mathrm{MFRMS} = 10 \times \log\left\{\frac{1}{M}\sum_{i=1}^{M}\left(\frac{A_f(i)}{A_{\mathrm{ref}}(i)}\right)^2\right\},\\ A_f(f \times S) = P_{\mathrm{AR}}(f),\\ 0 \leqslant f \leqslant \tfrac{1}{2},\end{gathered} \tag{12}$$

where $A_f(i)$ is the amplitude of the ith spectral response of the current spectrum, and $A_{\mathrm{ref}}(i)$ is the amplitude of the ith spectral response in the reference spectrum. M is the total number of lines in the spectrum, S is the sampling rate used in data acquisition, f is the fraction of the sampling rate, and $P_{\mathrm{AR}}(f)$ is the AR parametric spectrum calculated through the AR technique.

5. IMPLEMENTATION METHODOLOGY

5.1. *Introduction*

Several vibration imbalance conditions were generated using a rotating test rig. In order to study the capabilities of CMAC networks, a two-stage experimental procedure is proposed. First, a CMAC network is trained with different severity conditions. Then, a test data set is used to check the trained CMAC for severity prediction. Second, a CMAC network is trained by a set of different severity conditions, and is then given an untrained condition to test the CMACs ability to estimate the severity level of the unknown condition. The CMAC architecture used is a multiple-input one-output model. Figure 2 shows the diagram of the CMAC training and testing procedures.

5.2. *Experimental set-up*

Data collection in the form of vibration signals was conducted using the following setup: a test rig consisting of a d.c. motor connected to a shaft by a drivebelt with two pillow block bearings

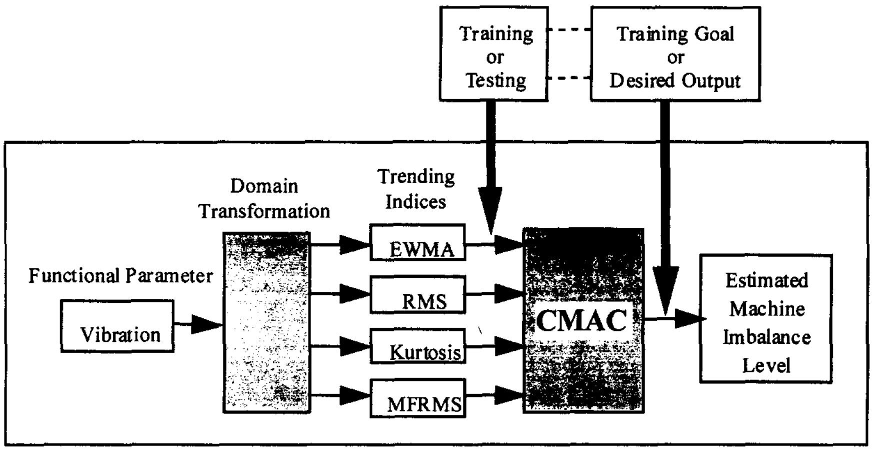

Fig. 2. CMAC training and testing procedures.

and two round hubs mounted on each end of the shaft; a steel plate was used to secure the whole test rig. Vibration signals were collected from the right bearing using a 328C04 PCB accelerometer mounted horizontally on the bearing housings. A DT2821-G-8DI data acquisition board was installed in a 486 IBM compatible PC. The entire setup is illustrated in Fig. 3.

5.3. Data preparation

In order to generate different imbalance levels (different machine fault severity levels), a piece of clay which was divided into six equal-weight parts was used. By adding a piece of clay at a time, the imbalance level of the test rig increased. Therefore, totally, there were seven different imbalance conditions labeled 1–7. Level 1 represented a balanced condition, meaning that there was no clay attached to the test rig. For each imbalance level, 15 signals were collected. Each signal was used to calculate one set of trending indices and AR parameters. The trending index set includes EWMA, Kurtosis, RMS and MFRMS. For the AR parametric model, an order of seven was chosen, which formed an AR(7) parameter vector containing seven AR parameters and one error term. In other

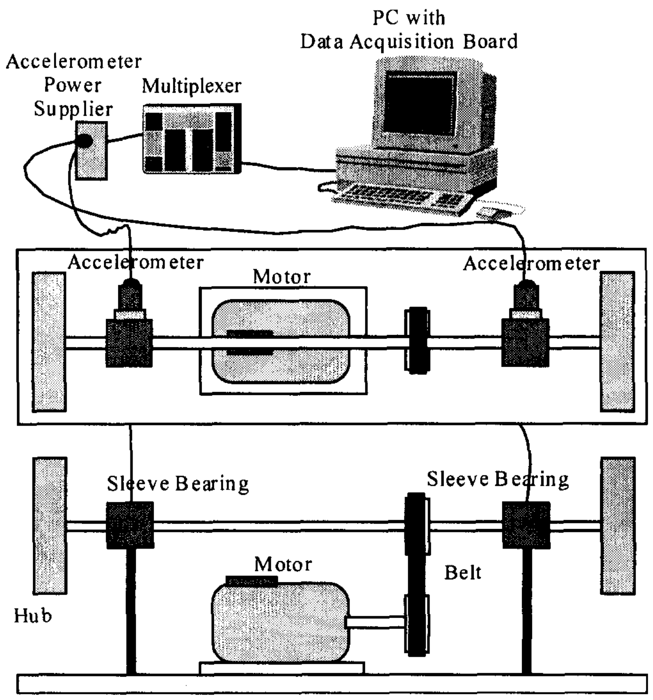

Fig. 3. The experimental setup for vibration signal acquisition.

words, 15 sets of trending indices and AR parameters were generated for each level. They were then used in the training and testing of CMACs.

The 15 data sets were divided into five groups. The average value of each group was recorded. For either the case of the trending index or the AR(7) parameter, each imbalance condition contained three training vectors and two test vectors. Also, since the CMAC is a supervised neural network, the level of the imbalance condition was assigned to each training vector as a training goal and to each test vector as a desired output. The prepared data sets are given in Appendix Tables A1 and A2, respectively.

5.4. Training sets, test sets, test results and test errors

In order to study the interpolation feature of the CMAC, three training sets, including different combinations of the training vectors, were designed. The X training set included all training vectors (three training vectors from each imbalance level, at a total of 21). For the Y training set, the training vectors from imbalance level 6 were intentionally left out. For the Z training set, the training vectors from imbalance levels 3 and 6 were left out. After each training section, the entire test set was still used to test the trained CMAC for interpolation property investigation.

A test result is the CMAC output after a test vector is entered. The test results of the test set can be written in the following vector format:

$$\mathbf{P} = \{p_1, p_2, \ldots, p_{14}\}, \tag{13}$$

where p_i is the ith test result. For each test result, there is an expected value associated with it. These expected values can be written as vector $\mathbf{P}_d$:

$$\mathbf{P}_d = \{p_{1d}, p_{2d}, \ldots, p_{14d}\}, \tag{14}$$

where p_{id} is the desired output of p_i.

A test error, TE, may be defined by computing the distance between vectors $\mathbf{P}$ and $\mathbf{P}_d$. TE is used as an accuracy index for each CMAC training section. Equation (15) is the equation of TE:

$$\mathrm{TE} = \sum_{i=1}^{14} \sqrt{(p_i - p_{id})^2}. \tag{15}$$

5.5. Training iteration

According to the CMAC learning algorithm, a small acceptable error ϵ must be given before the training procedure begins. In the training stage, the CMAC continues adjusting its weights until its prediction error is smaller than ϵ. However, in order to compare different CMAC configurations, a fixed number of training iterations was designed. For the trending indices inputs, 200 training iterations were used. For the AR(7) parameters case, 5000 training iterations were chosen.

5.6. Training with vibration trending indices

Figure 4 shows the normalized trending indices plot for all imbalance conditions. The EWMA, RMS and MFRMS indices in the plot show increasing trends from the imbalance level 1–7, but

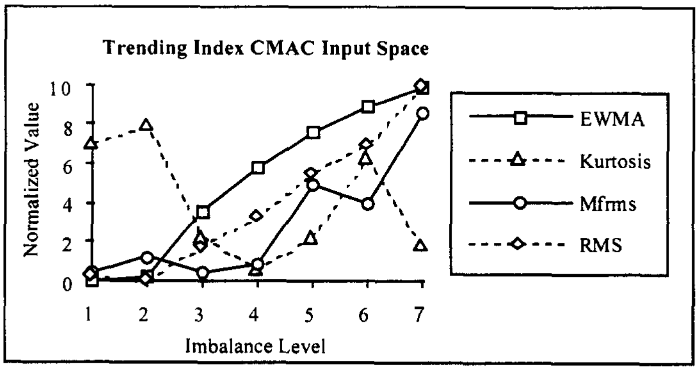

Fig. 4. Trending indices input space (normalized data sets).

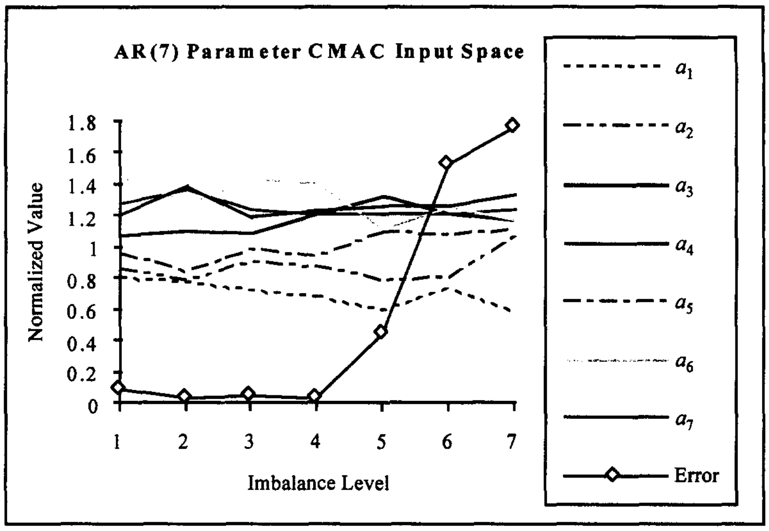

Fig. 5. AR(7) parameters input space (normalized data sets).

the Kurtosis index fluctuates considerably. Figure 4 basically shows the input space of the vibration trending indices case.

To date, there is no deterministic way of selecting a good set of parameters for the CMAC. In most cases the trial-and-error method is used. Usually, the first parameter to be determined is the physical memory size available on the computer. For the case of trending indices, the following CMAC parameters were specified:

$$N = 4, \quad R = 600, \quad |A_p| = 4096, \quad \beta = 1, \quad K = 32, 64, 128, 256, 512,$$

where five different K numbers were selected for five separate runs. The purpose was to test the generalization capability of the CMAC. Besides adjusting the K-value in the CMAC, three different training sets (X, Y and Z) were also used to investigate the CMAC's interpolation capability. The results are given in the next section.

5.7. Training with the AR(7) parameters case

Figure 5 shows the input space of the normalized AR(7) parameters. The seven AR(7) parameters and one error term are represented by $a_1, \ldots, a_7$, and σ^2, respectively. Figure 5 displays a more complicated situation for the CMAC training. The only visible trend is the error term σ^2. Due to the complexity of the AR(7) parameters input space only the X training set of the AR(7) parameters

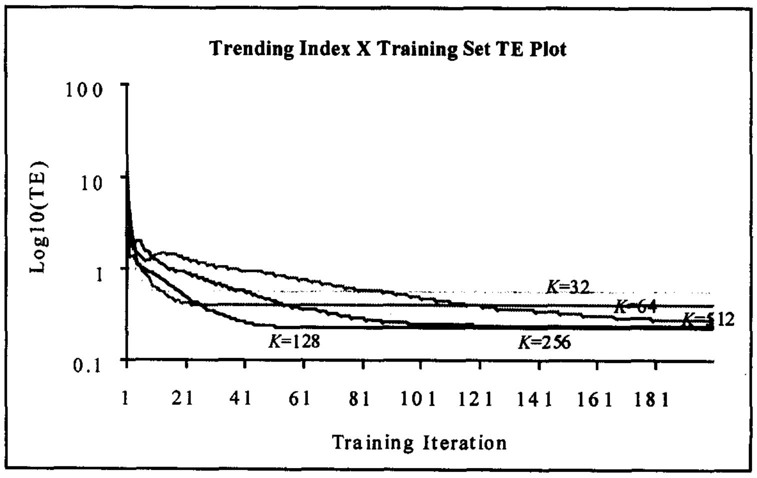

Fig. 6. Trending indices test error converging curves for different K numbers (training set = X; training iterations = 200).

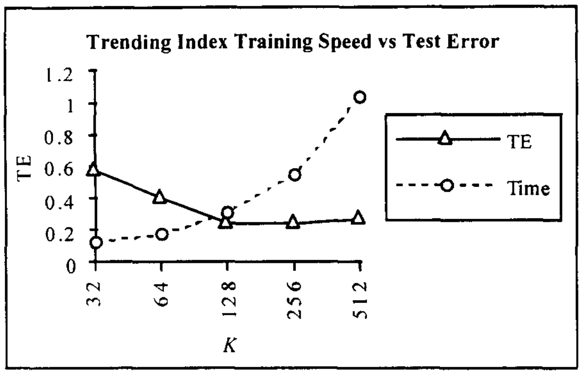

Fig. 7. Trending indices training speed vs test error for different K numbers (training set = X; training iterations = 200; time = 100 s).

with different K-values was implemented. Also, because of the ambiguous input–output relationship observed in the AR(7) parameters input space, the CMAC would need more training iterations. Therefore, the training iteration was set to 5000. Other CMAC parameters used are listed as follows:

$$N = 8, \quad R = 600, \quad |A_p| = 4096, \quad \beta = 1, \quad K = 32, 64, 128, 256, 512.$$

6. EXPERIMENTAL RESULTS AND DISCUSSION

6.1. Test results: vibration trending indices case

In this section the CMAC test results for the trending indices case are discussed. Figure 6 shows the TE curves when using the X training set with different K numbers. The logarithmic scale was applied on the Y-axis.

Note that the TE curves converged to steady states at different exponential rates for all K numbers. For example, in the case of $K = 128$, a steady state was reached after 40 learning iterations which took less than 1 s on a 486 PC. Also, the final test errors were all very small (from 0.56 to 0.24), indicating that the CMAC was able to estimate the machine imbalance levels correctly. Figure 7 illustrates how the CMAC generalization capability decreased (TE increased) as the number of quantizing function (K) decreased. However, there was a trade-off between the K-value and the calculation speed. Figure 7 shows that when the K-value increased, the CMAC network needed not only more memory, but also a longer calculation time.

The CMAC memory allocation problem has already been handled by hash-coding, which allows one to choose the maximum memory size available on the computer. Throughout the entire testing section, the memory size was fixed at 4096 × 32 bits. Therefore, the learning speed was the only factor influencing the selection of K. For the trending indices case, Fig. 7 suggests that $K = 128$ would be the best choice.

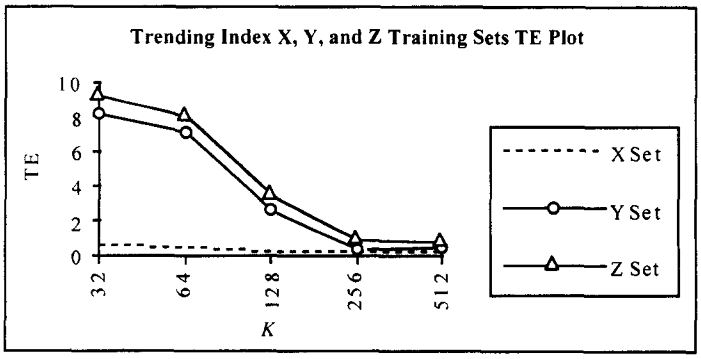

Fig. 8. Trending indices CMAC test errors for X, Y and Z training sets (Y set: level 6 missing; Z set: levels 6 and 3 missing; training iterations = 200).

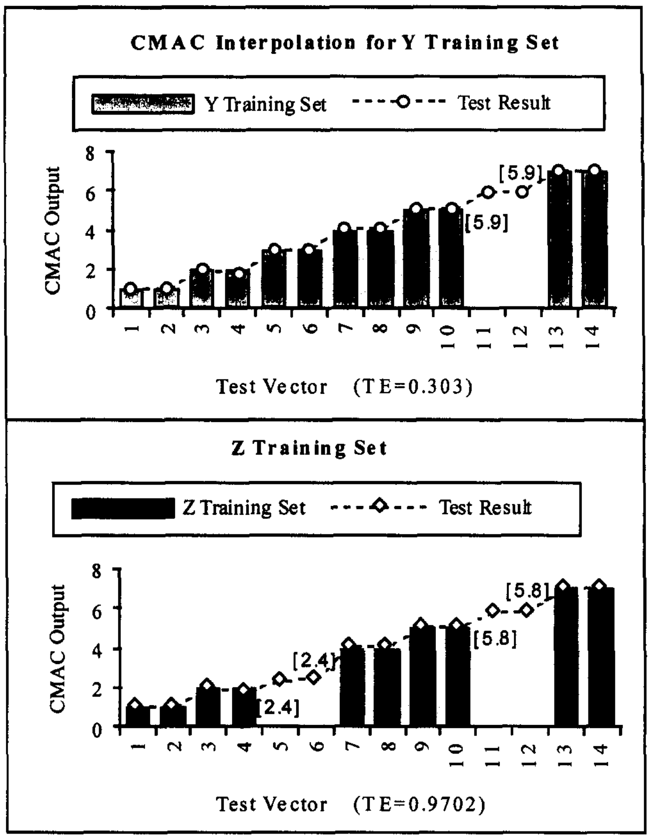

Fig. 9. Trending indices CMAC test results for Y and Z training sets ($K = 256$; training iterations = 200).

Figure 8 compares the test results from different training sets. The purpose is to investigate the interpolation capability of the CMAC network for the missing training vectors. Remember that both Y and Z training sets have training vectors missing. As shown in Fig. 8, the overall TE level of the X training set is much smaller than Y and Z, and Y is smaller than Z since it has only one training vector missing. In addition, the CMAC interpolation capability may be improved by increasing the K-value. That is, when a large K number is chosen (e.g. $K = 512$), the test errors for all training sets become very close, demonstrating the outstanding interpolation capability of the CMAC.

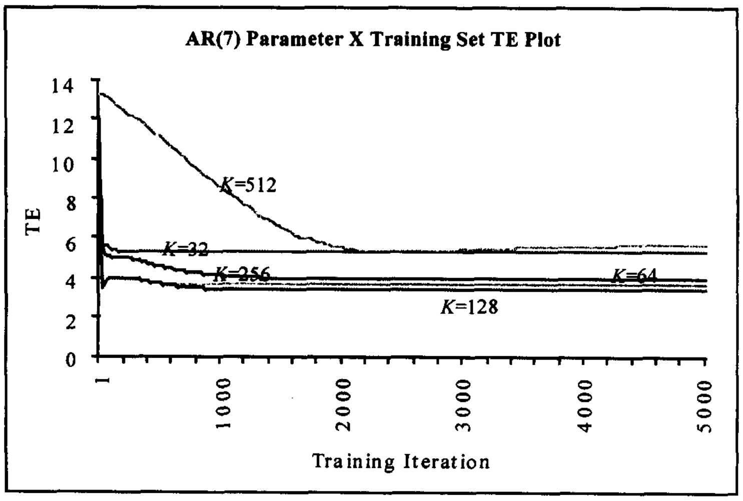

Fig. 10. AR(7) parameters test error curves for different K numbers (training set = X; training iterations = 5000).

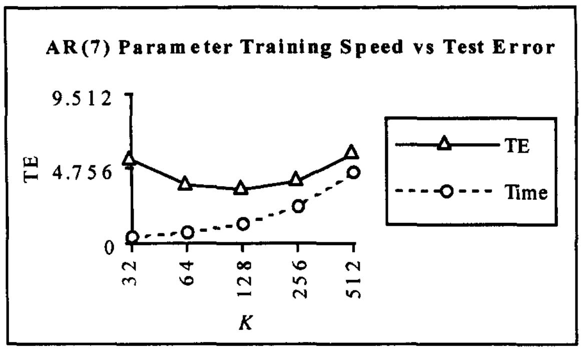

Fig. 11. AR(7) parameters training speed vs test error for different K numbers (training set = X; training iterations = 5000; time × 1000 s).

Figure 9 illustrates the detailed test results of using the Y and Z training sets. The solid bars represent the training vectors, and the dashed lines with circles or diamonds indicate the positions of CMAC test outputs. It shows that the CMAC network can predict the missed trained positions very well.

6.2. *Test results: AR(7) parameters case*

As expected, due to the complexity of the AR(7) parameters input space, the test results of the AR(7) parameters were not as reliable as those of the trending indices (cf. Fig. 10 to Fig. 6). However, three important phenomena can be seen from Fig. 10. First, the exponentially-decreasing training curve is highlighted. It might take a longer time to complete each training iteration, but the shapes of most curves remain similar. Second, when $K = 128$ the best test result was achieved, which is consistent to the trending indices case. Finally, the TE converging curve for $K = 512$ displayed an interesting "over-trained" phenomenon. That is, after about 2500 training iterations, the value of test errorreached its minimum and began climbing.

Figure 11 shows again as the K-value increased, the CMAC needed more calculation time. Obviously 128 is the most appropriate K-value since it has the minimum TE value.

Figure 12 demonstrates the detailed test results when the X training set of the AR(7) parameters was used. The solid bars represent the training vectors from the X training set, and the dashed line with circles shows the positions of the CMAC test outputs. It did not give a reliable estimation for each test vector. However, the dashed line did, in general, follow the trend of imbalance levels.

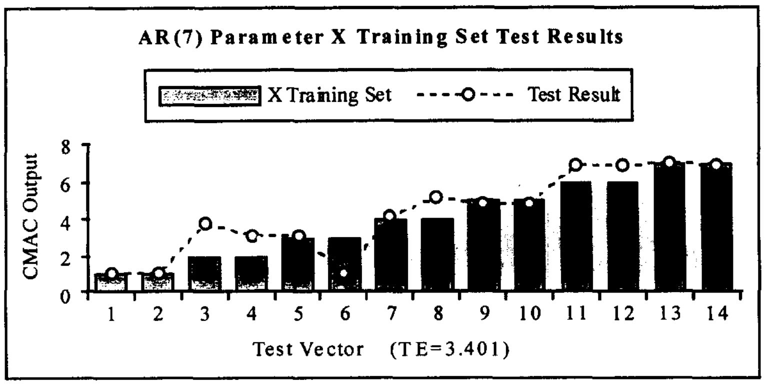

Fig. 12. AR(7) parameters CMAC test results for $K = 128$ (training set = X; training iterations = 5000).

7. CONCLUDING REMARKS

To meet the requirements of on-line machine fault monitoring and fault severity estimation in a predictive maintenance system, a new approach combining the CMAC neural network and an advanced trending monitoring method has been proposed and implemented in this paper.

Several potential capabilities of the CMAC network have been discussed and shown in the paper. The properties of the CMAC are notable. First, due to its outstanding functional interpolation and quantitatively descriptive feature, the CMAC may be used as a fault severity estimation tool. This overcomes the problem encountered in using other ANNs such as ART2, which performs only as a pattern classifier. Second, because of the local generalization property of the CMAC, incremental learning is no longer a problem. It used to be a major drawback of the backpropagation network. Finally, the exponential learning speed and great input–output generalization capability make on-line real-time machine performance estimation possible. The research recounted in this paper demonstrates that the selection of an adequate feature extraction scheme for the CMAC is also important. Using the combination of vibration trending indices as the CMAC input vectors outperforms using the AR parameters.

Currently, only one type of machine fault, rotating imbalance, has been studied. It is necessary to implement and validate the CMAC network with other faults or the combination of several faults before a complete predictive maintenance scheme can be implemented in real-world settings.

REFERENCES

1. S. D. Haddad. Condition monitoring and fault diagnosis for predictive maintenance and quality control. *Proc. 3rd Int. Machinery Monitoring and Diagnostics Conf.*, Las Vegas, NV, pp. 1–8 (1991).
2. S. Braun. *Mechanical Signature Analysis: Theory and Applications.* Academic Press, London (1986).
3. J. Mathew. Monitoring the vibrations of rotating machine elements—an overview. *ASME* **18-5,** 15–22 (1989).
4. N. Naruo, M. Lehto and G. Salvendy. Development of a knowledge-based decision support systems for diagnosing malfunctions of advanced production equipment. *Int. J. Product. Res.* **23,** 2259–2276 (1990).
5. M. A. Kramer. Malfunction diagnosis using quantitative models with non-Boolean reasoning in expert system. *AIChE Jl* **33,** 130 (1987).
6. B. Buchana and E. Shortliffe. *Rule-Based Expert Systems: the MYCIN Experiments of the Stanford Heuristic Programming Project.* Addison-Wesley, Reading, MA (1984).
7. R. P. Lippmann. Pattern classification using neural networks. *IEEE Commun. Mag.*, **Nov.** 47–64 (1989).
8. G. M. Knapp and H.-P. (B.) Wang. Machine fault classification: a neural network approach. *Int. J. Product. Res.* **30,** 811–823 (1992).
9. D. S. Kim, Y. S. Shin and D. K. Carlson. Machinery diagnostics for rotating machinery using back propagation neural network. *Proc. 3rd Int. Machinery Monitoring and Diagnostics Conf.*, Las Vegas, NV, pp. 309–320 (1991).
10. C.-C. Lin and H.-P. Wang. Classification of autoregressive spectral estimated signal patterns using an adaptive resonance theory neural network. *Computers Ind.* **22,** 143–158 (1993).
11. T. I. Liu and N. R. Iyer. Diagnosis of roller bearing defects using neural networks. *Int. J. Adv. Manufact. Technol.* **8,** 210–215 (1993).
12. W. T. Miller, F. H. Glanz and L. G. Kraft. CMAC: an associative neural network alternative to back propagation. *Proc. IEEE* **78,** 1561–1567 (1990).
13. J. S. Albus. A new approach to manipulator control: the cerebellar model articulation controller (CMAC). *Trans. ASMEJ. Dynam. Syst., Meas. Control* **97,** 220–227 (1975).
14. J. S. Albus. Data storage in the cerebellar model articulation controller (CMAC). *Trans. ASME J. Dynam. Syst., Meas. Control* **97,** 228–233 (1975).
15. J. S. Albus. *Brain, Behavior and Robotics.* BYTE Books, Peterborough, NH (1981).
16. Y. Wong and A. Sideris. Learning convergence in the cerebellar model articulation controller. *IEEE Trans. Neural Networks* **3,** 115–120 (1992).
17. J. Moody. Fast learning in multi-resolution hierarchies. *Advances in Neural Information Processing Systems 1* (D. S. Touretzky, Ed.), pp. 29–39. Morgan Kaufmann, Los Altos, CA (1989).
18. W. T. Miller. Real-time application of neural networks for sensor-based control of robots with vision. *IEEE Trans. Syst., Man Cybernet.* **19,** 825–831 (1989).
19. W. T. Miller, F. H. Glanz and L. G. Kraft. CMAC: an associative neural network alternative to back propagation. *Proc. IEEE* **78,** 1561–1567 (1990).
20. M. J. Carter, F. J. Rudolph and A. J. Nucci. Operational fault tolerance of CMAC networks. *Advances in Neural Information Processing Systems 2* (D. S. Touretzky, Ed.), pp. 340–347. Morgan Kaufmann, Los Altos, CA (1990).
21. D. J. Linse and R. F. Stengel. Neural networks for function approximation in nonlinear control. *Amer. Control. Conf.* Vol. 1, pp. 674–679 (1990).
22. J. Lee and B. M. Kramer. Analysis of machine degradation using a neural network based pattern discrimination model. *J. Manufact. Syst.* **12,** 379–387 (1993).
23. S. M. Kay and S. L. Marple Jr. Spectrum analysis—a modern perspective. *Proc. IEEE* **69,** 1380–1419 (1981).
24. W. Gersch and T. S. Liu. Time series methods for the synthesis of random vibration systems. *ASME, J. Appl. Mech.* **43,** 159–165 (1976).

25. D. N. Swingler. Frequency errors in MEM processing. *IEEE Trans. Acous. Speech, Signal Process.* **ASSP-28,** 257–259 (1980).
26. S. M. Kay. *Modern Spectral Estimation: Theory and Application.* Prentice-Hall, NJ (1988).
27. S. L. Marple Jr. *Digital Spectral Analysis with Applications.* Prentice Hall, NJ (1987).
28. D. Dyer and R. M. Stewart. Detection of rolling element bearing damage by statistical vibration analysis. *Trans. ASME J. Mech. Des.* **100,** 229–235 (1978).
29. J. Mathew and R. J. Alfredson. The condition monitoring of rolling element bearings using vibration analysis. *J. Vibrat. Acoust., Stress Reliab. Des.* **106,** 447–453 (1984).
30. J. K. Spoerre. Machine performance monitoring and fault classification using an exponentially weighted moving average scheme. Master's Thesis, Department of Industrial Engineering, University of Iowa (1993).
31. R. Monk. Vibration measurement gives early warning of mechanical faults. *Process Engng* **Nov,** 135–137 (1972).
32. C. K. Mechefske and J. Mathew. Trending the gradual deterioration of low speed rolling element bearings using parametric spectra. *Proc. 3rd Int. Machinery Monitoring and Diagnostics Conf.,* Las Vegas, NV, pp. 232–320 (1991).
33. J. P. Burg. Maximum entropy spectral analysis. *Proc. 37th Meeting of Soc. Exploration Geophysicists* (1967).

APPENDIX A

Table A1. The trending index CMAC input vectors

EWMA	Kurtosis	MFRMS	RMS	Goal
		Training vector		
0.04	6.88	0.40	0.17	1
0.05	7.51	0.51	0.18	1
0.02	7.27	0.39	0.17	1
0.20	7.11	1.35	0.04	2
0.18	7.96	1.45	0.03	2
0.23	8.54	1.10	0.08	2
3.58	1.73	0.20	1.71	3
3.61	1.80	0.31	1.71	3
3.64	2.36	0.55	1.76	3
5.71	0.54	0.52	3.18	4
5.71	0.37	1.09	3.23	4
5.71	0.39	0.96	3.26	4
7.60	1.82	4.98	5.42	5
7.61	1.68	5.24	5.49	5
7.62	2.22	4.82	5.50	5
8.92	4.68	3.78	6.81	6
8.97	6.67	4.00	7.00	6
9.02	6.57	3.84	7.00	6
9.81	1.46	9.43	9.82	7
9.94	1.87	8.66	9.94	7
9.94	1.79	8.05	9.93	7
		Test vector		
0.02	6.43	0.43	0.18	1
0.05	6.88	0.46	0.20	1
0.21	7.92	1.12	0.06	2
0.22	7.96	0.98	0.09	2
3.62	2.32	0.34	1.73	3
3.63	2.71	0.70	1.77	3
5.76	0.48	0.50	3.35	4
5.72	0.65	0.74	3.30	4
7.59	2.52	4.54	5.53	5
7.59	2.53	5.28	5.54	5
8.92	7.04	4.13	6.97	6
8.90	6.47	4.25	6.92	6
9.94	1.90	9.12	9.90	7
9.99	2.08	7.93	9.81	7

(*Appendix continued overleaf*)

Table A2. The AR(7) parameter CMAC input vectors

a_1	a_2	a_3	a_4	a_5	a_6	a_7	Error	Goal
				Training vector				
0.81	0.88	1.05	1.28	0.98	1.45	1.22	0.08	1
0.79	0.85	1.07	1.30	0.96	1.44	1.20	0.09	1
0.81	0.85	1.07	1.30	0.95	1.46	1.20	0.07	1
0.76	0.80	1.11	1.34	0.84	1.28	1.40	0.03	2
0.75	0.83	1.07	1.37	0.82	1.30	1.39	0.02	2
0.79	0.76	1.12	1.37	0.84	1.27	1.39	0.05	2
0.76	0.85	1.14	1.24	0.94	1.45	1.22	0.02	3
0.73	0.89	1.06	1.25	0.99	1.46	1.16	0.04	3
0.70	0.90	1.11	1.27	0.98	1.38	1.22	0.06	3
0.67	0.88	1.23	1.20	0.92	1.42	1.23	0.04	4
0.66	0.89	1.23	1.21	0.92	1.40	1.24	0.03	4
0.68	0.87	1.20	1.23	0.93	1.42	1.21	0.03	4
0.64	0.76	1.34	1.22	1.07	1.14	1.28	0.37	5
0.61	0.80	1.30	1.22	1.11	1.12	1.26	0.40	5
0.60	0.79	1.35	1.22	1.06	1.13	1.28	0.41	5
0.71	0.80	1.21	1.22	1.06	1.25	1.26	1.15	6
0.73	0.81	1.23	1.21	1.06	1.27	1.27	1.59	6
0.72	0.83	1.21	1.21	1.05	1.28	1.27	1.48	6
0.58	1.04	1.19	1.24	1.11	1.16	1.32	1.64	7
0.59	1.11	1.13	1.25	1.13	1.17	1.34	1.83	7
0.61	1.07	1.17	1.27	1.06	1.21	1.34	1.79	7
				Test vector				
0.78	0.87	1.09	1.29	0.93	1.46	1.21	0.08	1
0.79	0.84	1.13	1.29	0.93	1.45	1.22	0.08	1
0.80	0.78	1.09	1.36	0.85	1.31	1.38	0.03	2
0.79	0.77	1.11	1.38	0.85	1.28	1.38	0.05	2
0.71	0.91	1.12	1.22	0.98	1.44	1.19	0.05	3
0.72	0.93	1.06	1.26	1.00	1.43	1.18	0.08	3
0.72	0.84	1.22	1.22	0.95	1.42	1.22	0.03	4
0.67	0.89	1.21	1.22	0.96	1.38	1.23	0.04	4
0.57	0.82	1.30	1.20	1.13	1.11	1.24	0.52	5
0.56	0.81	1.32	1.22	1.09	1.10	1.25	0.51	5
0.76	0.76	1.21	1.24	1.08	1.25	1.25	1.69	6
0.74	0.80	1.19	1.23	1.10	1.23	1.26	1.68	6
0.56	1.10	1.16	1.23	1.11	1.18	1.32	1.81	7
0.58	0.99	1.22	1.23	1.12	1.14	1.30	1.73	7