Available online at www.sciencedirect.com
SCIENCE DIRECT®
Journal of Sound and Vibration 285 (2005) 1–25
JOURNAL OF SOUND AND VIBRATION
www.elsevier.com/locate/jsvi

Analysis and DSP implementation of an ANC system using a filtered-error neural network

Ya-Li Zhou[a,*], Qi-Zhi Zhang[a], Xiao-Dong Li[b], Woon-Seng Gan[c]

[a]*Department of Computer Science and Automation, Beijing Institute of Machinery, P.O. Box 2865, Beijing 100085, People's Republic of China*
[b]*Institute of Acoustic, Academia Sinica, Beijing 100080, People's Republic of China*
[c]*School of Electronics and Electrical Engineering, Nanyang Technological University, Singapore 639798, Singapore*

Received 16 October 2003; received in revised form 8 June 2004; accepted 11 August 2004
Available online 23 November 2004

Abstract

In this paper, feedforward active noise control (ANC) using a neural network (NN) based on filtered-error back-propagation (BP) algorithm is considered. The filtered-error BP NN (FEBPNN) algorithm is first derived, and the difference between the FEBPNN algorithm and the filtered-X BP NN (FXBPNN) algorithm is given to show that the FEBPNN algorithm offers computational advantage over the FXBPNN algorithm. Computer simulations are carried out to compare the FEBPNN algorithm with the filtered-X least mean square (FXLMS) algorithm and the FXBPNN algorithm. The controllers based on the FEBPNN algorithm and the FXLMS algorithm are implemented on a Texas Instruments digital signal processor (DSP) TMS320VC33. The simulations and the experimental verification tests show that the FEBPNN algorithm performs as well as the FXLMS algorithm for a linear control problem, and better for a nonlinear control problem, at the same time, the simulations and the experimental verification tests also show that the convergence rate of the FEBPNN is acceptable, and the FEBPNN has better tracking ability under changes of the primal signal, the primary path and the secondary path. The experiments also lead to the conclusion that more work is required to improve the predictability and consistency of the performance of the NN controller based on the FEBPNN algorithm.

*Corresponding author.
E-mail address: zhouyali@yahoo.com (Y.-L. Zhou).

0022-460X/$ - see front matter © 2004 Elsevier Ltd. All rights reserved.
doi:10.1016/j.jsv.2004.08.007

1. Introduction

Based on the rapid progress of high-speed and low-cost computing devices such as digital signal processor (DSP), digital active noise control (ANC) techniques have been receiving much attention because of the better performance over conventional passive methods in attenuating low-frequency noises [1]. A broadband feedforward ANC system in a duct is shown in Fig. 1 [2], where a reference microphone is used to pick up the reference input signal. The reference input signal $x(n)$ is in turn processed by the ANC system to generate the control signal $y(n)$ of equal amplitude but 180° out of phase from the reference input signal. The control signal is used to drive the canceling loudspeaker to attenuate the primary acoustic noise in the duct. The error signal $e(n)$ is used to adjust the controller parameters. The most common form of adaptive algorithm/architecture combination is a transversal finite impulse response (FIR) filter using the filtered-X least mean square (FXLMS) algorithm [1], which has been widely used in a variety of practical applications over the past decades. But, such a system is only limited to linear control problems. When the ANC system comprises nonlinear components (such as, when the control actuator exhibits nonlinear response characteristics, the preamplifier is in saturation, or the reference input signal is distorted, etc.), the linear controller will not work well.

In these cases, a nonlinear adaptive control scheme may prove superior to a linear one. The feedforward neural network (NN) is a perfect control technology for nonlinear systems, and has been used in various systems with nonlinearity [3]. For active control of sound and vibration, the feedforward NN based on back-propagation (BP) algorithm has been investigated by several researchers and experimental investigations have been made [4–7]. It would seem likely that such an architecture/algorithm combination would be employed to perform the previously mentioned nonlinear active control tasks, where the NN would be trained to derive an output signal which would cancel the noise. Of course, owing to the inclusion of the tapped delay line input to the transfer function model, the standard BP algorithm cannot be used directly in the ANC system, so it must be modified to enable adaptation of the NN controller for use in feedforward ANC systems [4].

The back-propagation NN (BPNN) algorithm is modified to be appropriate for ANC systems. The reference signal is filtered by the estimate of the secondary path transfer function, and since the reference signal is generally denoted as x, so in this paper, it is referred to as filtered-X BPNN (FXBPNN) algorithm. One drawback to this algorithm is that it has relatively high computational load. In practical implementations, this means that a higher clock rate DSP is required, especially when a higher sampling frequency is required or the NN is required to

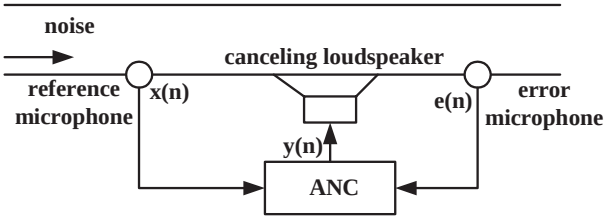

Fig. 1. Feedforward ANC system in a duct.

expand. To solve the above problem, a filtered-error BPNN (FEBPNN) algorithm is proposed here. As the term suggests, instead of reference signal $x(n)$, the error signal $e(n)$ is filtered by the estimate of the secondary path transfer function. It can reduce the computational load greatly, which will be discussed in detail in this paper.

This paper is organized as follows. Section 2 shows the details of the FEBPNN algorithm for implementing a feedforward NN in the ANC system. The simulation results in using the proposed FEBPNN algorithm and other conventional algorithms in canceling noise over nonlinear primary noise path are shown in Section 3. This is followed by the experimental testing of the ANC system based on FEBPNN algorithm and comparison with the FXLMS algorithm in Section 4. The results show that the FEBPNN algorithm is effective for nonlinear control problems. Finally, a qualitative discussion of the performance of the FEBPNN implementation in the feedforward ANC system is discussed in Section 5, and concludes in Section 6.

2. BPNN algorithms development

The block diagram of a feedforward ANC system using the NN algorithm and the NN controller with a single hidden layer are shown in Figs. 2 and 3, respectively. The primary path, $P(Z)$ is from the reference microphone to the error microphone, the secondary path, $S(Z)$ is from the canceling loudspeaker to the error microphone.

It is obvious that the existence of the secondary path transfer function $S(Z)$ must be taken into account when adapting the control system if adaptive algorithm stability is to be maintained [4]. For simplicity, assuming that the secondary path is linear and time-invariant, it is a usual practice to identify the secondary path off-line during a training stage. During the active canceling mode (on-line), a fixed transfer function, $H(Z)$ (where $H(Z) = \hat{S}(Z)$) is used. The secondary path identification is introduced in Ref. [8], where the secondary path transfer function is modeled as a FIR filter. The objective of this section is to develop adaptive algorithms based on the BPNN to improve the noise cancellation capability of a nonlinear system. Two algorithms, namely, FXBPNN and FEBPNN will be described in the following sections.

2.1. FXBPNN algorithm

In order to compare the FEBPNN algorithm with the FXBPNN algorithm, the FXBPNN algorithm will first be derived, in fact, the FXBPNN is simply an extension of the linear FXLMS

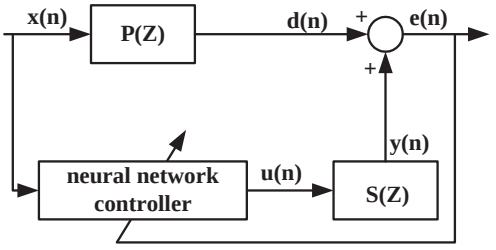

Fig. 2. Block diagram of ANC system using the NN algorithm.

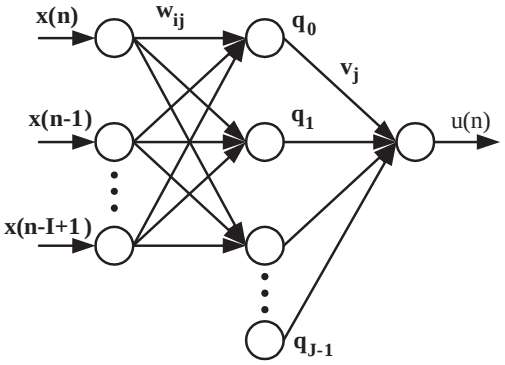

Fig. 3. The NN controller.

algorithm as having only a single output layer with a single node, having a linear nodal output function [4], it is also a form of gradient descent algorithm in which the error signal is used to adjust the weights in the output and hidden layers. The error is back-propagated from the output to the input of the NN to adjust its weights.

The error criterion of the controller is to minimize the mean square value of the error signal $e(n)$:

$$\min E[e^2(n)]. \tag{1}$$

For practicality, the instantaneous estimate of squared error is used to approximate the mean square value of the error signal $e(n)$, so that the error criterion actually employed in the algorithm derivation is [4]:

$$\min e^2(n). \tag{2}$$

The FXBPNN algorithm is implemented in two stages, (A) forward and (B) backward propagation as follows:

(*A*) *Forward propagation*:

Step (1): Input the reference signal $X(n)$ and the error signal $e(n)$.

Where $X(n) = [x(n)\, x(n-1) \ldots x(n-(I-1))]^{\mathrm{T}}$ is the vector of reference signal samples at time n, $e(n)$ is the error signal at time n, and I is the number of the input layer neurons.

Step (2): Calculate the output of the hidden layer $q_j(n)$.

$$q_j(n) = f_j[\mathrm{net}_j(n)], \tag{3}$$

where $\mathrm{net}_j(n) = \sum_{i=0}^{I-1} w_{ij}(n)x(n-i)$.

$$W_{IJ}(n) = \begin{bmatrix} w_{00}(n) & w_{01}(n) & w_{0J-2}(n) \\ w_{10}(n) & w_{11}(n) & w_{1J-2}(n) \\ w_{I-10}(n) & w_{I-11}(n) & w_{I-1J-2}(n) \end{bmatrix}$$

is the matrix of weights in the hidden layer.

$q_j(n)$ is the output of the jth hidden layer neuron.

$f(.)$ is a smoothing nonlinear activation function.

$$q_{J-1}(n) = -1.$$

$$j = 0, 1, \ldots, J-2.$$

J is the number of the hidden layer neurons.

Step (3): Calculate the output of the output layer $u(n)$.

$$u(n) = \sum_{j=0}^{J-1} [v_j(n) q_j(n)] = V^0(n) Q(n), \tag{4}$$

where $u(n)$ is the output of the output layer at time n.

$V^0(n) = [v_0(n)\ v_1(n)\ \ldots\ v_{J-1}(n)]$ is the vector of weights in the output layer.

$Q(n) = [q_0(n)\ q_1(n)\ \ldots\ q_{J-1}(n)]^{\mathrm{T}}$ is the vector of the output of the hidden layer.

Step (4): Calculate the output of the system $y(n)$.

$$y(n) = \sum_{j=0}^{M-1} s_j(n) u(n-j) = S(n) U(n), \tag{5}$$

where $S(n) = [s_0(n)\ s_1(n)\ \ldots\ s_{M-1}(n)]$ is the secondary path model, whose z-transform is $S(z)$.

$U(n) = [u(n)\, u(n-1)\ \ldots\ u(n-(M-1))]^{\mathrm{T}}$ is the vector of the output of the output layer.

M is the length of the secondary path.

(*B*) *Back propagation*:

Using Eq. (2), the cost function is defined as

$$J(n) = \tfrac{1}{2} e^2(n) = \tfrac{1}{2} [y(n) + d(n)]^2. \tag{6}$$

The gradient descent algorithm is employed to adjust the weights of the control system. The weights are updated by adding a negative gradient of the error criterion with respect to the weights of interest

$$W(n+1) = W(n) - \mu \Delta W(n), \tag{7}$$

where μ is the learning rate (step size).

Noting that the primary disturbance $d(n)$ is in no way a function of the weights of the control system, the gradient estimate used in the adaptive algorithm is

$$\Delta W(n) = \frac{\partial J(n)}{\partial W(n)} = e(n) \frac{\partial y(n)}{\partial W(n)}. \tag{8}$$

The aim of the following analysis, based upon the preceding framework, is to find a solution to Eq. (8) for each weight in the NN controller. The solution can then be substituted into the gradient descent algorithm of Eq. (7) to adapt each weight in the control system.

Step (1): Update the weights in the output layer of the NN controller $v_j(n)$ as

$$\frac{\partial y(n)}{\partial v_j(n)} = \sum_{i=0}^{M-1} h_i \frac{\partial y(n-i)}{\partial v_j(n)}, \quad j = 0, 1, \ldots, J-1, \tag{9}$$

where $H = [h_0\, h_1 \ldots h_{M-1}]$ is the Mth-order impulse response of the secondary path model, whose z-transform is $H(z)$, which is estimated during training stage prior to the start of the ANC.

Assuming that the weights of the NN controller $v_j(n)$ are made to adapt slowly with time, then the following approximate estimation is used:

$$\frac{\partial u(n-i)}{\partial v_j(n)} \approx \frac{\partial u(n-i)}{\partial v_j(n-i)}. \tag{10}$$

Such that

$$\frac{\partial u(n-i)}{\partial v_j(n)} \approx \frac{\partial u(n-i)}{\partial v_j(n-i)} = \frac{\partial [V^0(n-i)Q(n-i)]}{\partial v_j(n-i)} = q_j(n-i). \tag{11}$$

Substituting Eq. (11) into Eq. (9) results in

$$\frac{\partial y(n)}{\partial v_j(n)} = \sum_{i=0}^{M-1} h_i q_j(n-i). \tag{12}$$

Thus the adaptation algorithm of the weights in the output layer can be written as

$$v_j(n+1) = v_j(n) - \mu e(n) \sum_{i=0}^{M-1} h_i q_j(n-i), \quad j = 0, 1, \ldots, J-1. \tag{13}$$

Step (2): Update the weights in the hidden layer of the NN controller $W_{IJ}(n)$.

$$\frac{\partial y(n)}{\partial w_{ij}(n)} = \sum_{k=0}^{M-1} h_k \frac{\partial u(n-k)}{\partial w_{ij}(n)}, \quad i = 0, 1, \ldots, I-1, \quad j = 0, 1, \ldots, J-2. \tag{14}$$

Assuming that the weights of the NN $W_{IJ}(n)$ are made to adapt slowly with time, the following approximate estimation is used:

$$\frac{\partial u(n-k)}{\partial w_{ij}(n)} \approx \frac{\partial u(n-k)}{\partial w_{ij}(n-k)}. \tag{15}$$

Noting that according to Fig. 3, the weights in the output layer $v_j(n-k)$ is in no way a function of the weights in the hidden layer, so

$$\frac{\partial u(n-k)}{\partial w_{ij}(n)} \approx \frac{\partial u(n-k)}{\partial w_{ij}(n-k)} = \frac{\partial [V^0(n-k)Q(n-k)]}{\partial w_{ij}(n-k)} = v_j(n-k) \frac{\partial q_j(n-k)}{\partial w_{ij}(n-k)}. \tag{16}$$

Recalling that $q_j(n) = f_j[\mathrm{net}_j(n)]$ then

$$\frac{\partial q_j(n-k)}{\partial w_{ij}(n-k)} = f_j'(\mathrm{net}_j(n-k)) \frac{\partial \mathrm{net}_j(n-k)}{\partial w_{ij}(n-k)}, \tag{17}$$

$$\frac{\partial \mathrm{net}_j(n-k)}{\partial w_{ij}(n-k)} = x(n-i-k). \tag{18}$$

Substituting Eqs. (16), (17) and (18) into Eq. (14) results in

$$\frac{\partial y(n)}{\partial w_{ij}(n)} = \sum_{k=0}^{M-1} h_k v_j(n-k) f_j'(\mathrm{net}_j(n-k)) x(n-i-k). \tag{19}$$

Thus the adaptation algorithm of the weights in the hidden layer can be written as

$$w_{ij}(n+1) = w_{ij}(n) - \mu e(n) \sum_{k=0}^{M-1} h_k v_j(n-k) f_j'(\mathrm{net}_j(n-k)) x(n-i-k),$$

$$i = 0, 1, \ldots, I-1, \quad j = 0, 1, \ldots, J-2. \tag{20}$$

From Eqs. (13) and (20), it can be seen that in the feedforward ANC system, the reference signal (for the hidden layer, the reference signal is $x(n-i)$, for the output layer, the reference signal is $q_j(n)$) is filtered by the estimate of the secondary path transfer function H (i.e., the reference signal is convolved with h_j). Therefore, this algorithm is called the filtered-reference BPNN algorithm, or the FXBPNN algorithm.

2.2. FEBPNN algorithm

In contrast to the FXBPNN algorithm, the FEBPNN algorithm takes into account of the presence of the plant response in the feedforward control problem by filtering the error signal, instead of filtering the reference signal. The filtered-error LMS algorithm for a linear system has been derived by Elliott [5], on the basis of this, the FEBPNN algorithm is derived in the following analysis. Since only the formulae that are used to update the weights are different between the FXBPNN algorithm and the FEBPNN algorithm, so only the weight-updating equations of the FEBPNN algorithm are given here.

Step (1): Update the weights in the output layer of the NN $v_j(n)$.

Using Eqs. (8), (12), then

$$\Delta v_j(n) = e(n) \sum_{i=0}^{M-1} h_i q_j(n-i). \tag{21}$$

Recalling Eq. (1), then the derivative of the time-averaged square error with respect to the jth weight in the output layer can be written as

$$\lim_{N\to\infty} \frac{1}{2N} \sum_{n=-N}^{N} \frac{\partial J(n)}{\partial v_j(n)} = \lim_{N\to\infty} \frac{1}{2N} \sum_{n=-N}^{N} \left[e(n) \sum_{i=0}^{M-1} h_i q_j(n-i) \right]. \tag{22}$$

It is also possible to rewritten Eq. (22) in an alternative form by defining a dummy time variable

$$k = n - i \quad \text{so that } n = k + i. \tag{23}$$

Eq. (22) can now be expressed as

$$\lim_{N\to\infty} \frac{1}{2N} \sum_{n=-N}^{N} \frac{\partial J(n)}{\partial v_j(n)} = \lim_{N\to\infty} \frac{1}{2N} \sum_{i=0}^{M-1} \sum_{k+i=-N}^{N} q_j(k) h_i e(k+i). \tag{24}$$

The signal $f(k)$ that would be obtained by noncausally filtering the error signal with the time-reversed impulse response of the plant is now defined to be

$$f(k) = \sum_{i=0}^{M-1} h_i e(k+i). \tag{25}$$

It is also noted that because i is always finite, the summation from $-N$ to N as N tends to ∞ on the right-hand side of Eq. (24) will be the same for $k = -\infty$ to ∞ as for $k+i = -\infty$ to ∞. The time-averaged derivative in Eq. (24) can thus be written as

$$\lim_{N\to\infty} \frac{1}{2N} \sum_{n=-N}^{N} \frac{\partial J(n)}{\partial v_j(n)} = \lim_{N\to\infty} \frac{1}{2N} \sum_{n=-N}^{N} f(n) q_j(n). \tag{26}$$

The problem with using Eq. (26) in a real-time algorithm is that an instantaneous estimate of $f(n)$ in Eq. (25) cannot be implemented with a causal system. This problem can be overcome by delaying both $f(n)$ and $q_j(n)$ in Eq. (26) by $M-1$ samples. The final form of the FEBPNN algorithm for the output layer is obtained by adapting the controller's coefficients with an instantaneous version of the derivative given by Eq. (26) delayed by $M-1$ samples, which can be written as

$$v_j(n+1) = v_j(n) - \mu f(n-M+1) q_j(n-M+1). \tag{27}$$

The delayed filtered-error signal, which would be used in practice, is equal to

$$f(n-M+1) = \sum_{i=0}^{M-1} h_i e(n+i-M+1). \tag{28}$$

Let $i' = M-1-i$, then

$$f(n-M+1) = \sum_{i'=0}^{M-1} h_{M-1-i'} e(n-i'). \tag{29}$$

The error signal is now causally filtered using a time-reversed version of the secondary path impulse response. Therefore, the weights in the output layer of the FEBPNN algorithm can be updated as shown:

$$v_j(n+1) = v_j(n) - \mu q_j(n-M+1) \sum_{i=0}^{M-1} h_{M-1-i} e(n-i), \quad j = 0, 1, \ldots, J-1. \tag{30}$$

Step (2): Update the weights in the hidden layer of the NN $W_{IJ}(n)$.

Using the similar method as in Eq. (20), we can derive the formula for updating the weights in the hidden layer of the NN $W_{IJ}(n)$ as

$$w_{ij}(n+1) = w_{ij}(n) - \mu v_j(n-M+1) f_j'(\mathrm{net}_j(n-M+1)) x(n-i-M+1) \sum_{k=0}^{M-1} h_{M-1-k} e(n-k),$$

$$i = 0, 1, \ldots, I-1, \quad j = 0, 1, \ldots, J-2. \tag{31}$$

Then the weights in the hidden layer of the FEBPNN algorithm can be updated by using Eq. (31).

If the z-transform of the secondary path model is written as

$$H(z) = \sum_{i=0}^{M-1} h_i z^{-i}. \tag{32}$$

Then the transfer function of the filter required to generate the delayed filtered error signal $f(n-M+1)$ from the error signal $e(n)$ can be written as

$$z^{-M+1} H(z^{-1}) = \sum_{i=0}^{M-1} h_i z^{i-1-M}. \tag{33}$$

Fig. 4 shows the block diagram of the complete FEBPNN algorithm.

2.3. Comparison of the two algorithms

It is necessary to compare the two algorithms to show that the FEBPNN algorithm offer computational advantage over the FXBPNN algorithm. It is obvious that for the FEBPNN algorithm, the M (the length of the secondary path) filtered error signals are involved in Eqs. (30) and (31). Considering the case with I input layer nodes and the J hidden layer nodes, the number of the multiplication operations required per sample to update the weights in the output layer is about $M+J$, and the number of the multiplication operations required per sample to update the weights in the hidden layer is about $M+(2+I)(J-1)$. In contrast, for the FXBPNN algorithm using Eqs. (13) and (20), the number of the multiplication operations required per sample to update the weights in the output layer is about $JM+J$, and the number of the multiplication operations required per sample to update the weights in hidden layer is about $(3M+1)(J-1)I$. For example when using $M=256$, $J=15$ and $I=25$, the FXBPNN and FEBPNN algorithms require about 2 73 005 and 905 multiplications per sample for its implementation, respectively. So in terms of their computational efficiency, the FEBPNN algorithm is far superior to the FXBPNN algorithm.

In the limit of very slow adaptation, the conditions to ensure stability for the FEBPNN are similar to that of the FXBPNN algorithm [5]. The details were reported in Ref. [9], and only the result is given here.

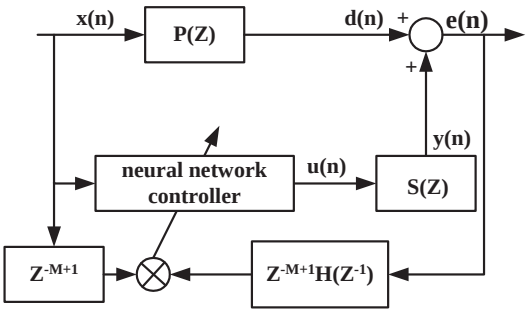

Fig. 4. Block diagram of the FEBPNN algorithm.

Theorem. *Let μ be the learning rate for the weights of the FEBPNN, we define $g_{\max} = \max_n \|G(n)\|$, $g_0 = \|H\|$, where $\|\cdot\|$ is the norm of matrix or vector, $H = [h_0, h_1, \ldots, h_{M-1}]$ is the coefficients of the secondary path model, $G(n) = \partial U(n)/\partial \overline{W}(n)$, $\overline{W} = [W^{\mathrm{T}}, V^{\mathrm{T}}]$ is weight matrix. If the learning rate μ is chosen as $0 < \mu < 2/(g_0 g_{\max})^2$, then the local stability of closed-loop control system based on the FEBPNN is guaranteed.*

It is obvious that the weights are updated along negative gradient direction of the error, so if the learning rate is small enough, the error would descend.

3. Simulation examples

Some simulations are presented to illustrate the properties of the FEBPNN algorithm, and at the same time, a comparison between the FXBPNN algorithm, the FEBPNN algorithm, and the FXLMS algorithm is made. The sampling rate used in this simulation is 1000 Hz. A 50 Hz sinusoidal signal is superimposed onto a white noise signal, which is used to generate the primary disturbance signal via second-order low-pass filter, with a cutoff frequency of 350 Hz, and also to generate the reference signal available to the control algorithm. A 16-tap FIR filter is used in the FXLMS algorithm, the number of neurons in the FEBPNN algorithm and the FXBPNN algorithm is set as 10-4-1.

Case 1: A linear ANC example is first considered to illustrate the FEBPNN algorithm effectiveness by comparison with the results given by others. The acoustic paths are chosen as follows:

The primary acoustic path from noise source to error microphone is [9]

$$d(n+1) = 0.8x(n-6) + 0.6x(n-7) - 0.2x(n-8) - 0.5x(n-9) - 0.1x(n-10) + 0.4x(n-11) - 0.05x(n-12).$$

The secondary acoustic path from secondary source to error microphone is [9]

$$y(n+1) = 0.9u(n-2) + 0.6u(n-3) + 0.1u(n-4) - 0.4u(n-5) - 0.1u(n-6) + 0.2u(n-7) + 0.1u(n-8) + 0.01u(n-9) + 0.001u(n-10).$$

Fig. 5 gives the simulation result of the canceling errors in the frequency domain. The upper curve shows the power spectrum of active noise canceling error when the ANC system turns off, and the dashed-line curve shows the power spectrum of active noise canceling error when the FXLMS algorithm is used to adapt the coefficients of the controller. The thin solid line shows the power spectrum of active noise canceling error when the FXBPNN algorithm is used to adapt the coefficients of the controller, while the thick solid line shows the power spectrum of active noise canceling error when the FEBPNN algorithm is used to adapt the coefficients of the controller. From the results shown in Fig. 5, it can be seen that the three algorithms have a similar performance for a linear control problem. But, to achieve the similar canceling error, the number of iterations required by the FEBPNN and the FXBPNN algorithms is twice as long as in the FXLMS algorithm (in this case, the number of iterations for the FEBPNN algorithm and the FXBPNN algorithm is 20 000, while the number of iterations for the FXLMS algorithm is 10 000).

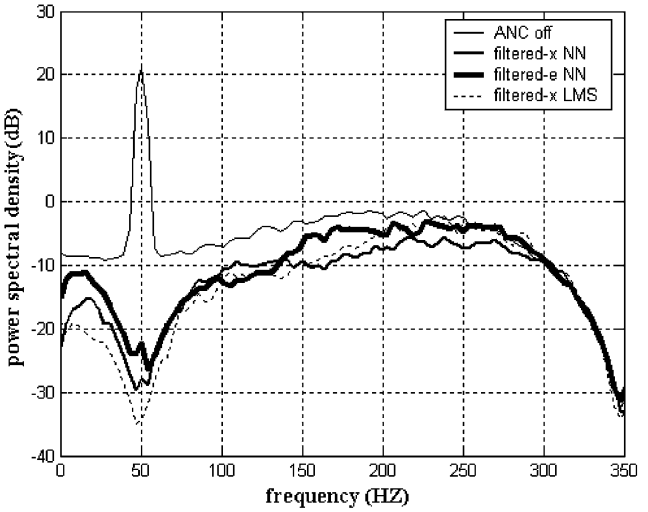

Fig. 5. Power spectrum of active noise canceling errors for linear case.

Case 2: An ANC example with nonlinear primary acoustic path is selected to illustrate the effectiveness of the three algorithms. The acoustic path is chosen as follows:

The primary acoustic path from noise source to error microphone is

$$d(n+1) = 0.8x(n-6) + 0.6x(n-7) - 0.2x(n-8) - 0.5x(n-9) - 0.1x(n-10) + 0.4x(n-11) - 0.05x(n-12) + 2.5x^3(n-6).$$

The secondary acoustic path from secondary source to error microphone is the same as that for linear case.

Fig. 6 gives the simulation result of the canceling errors in the frequency domain. The number of iterations is 18 000. The same line notations as in the previous case are used here. From the results shown in Fig. 6, it can be seen that all the three algorithms can reduce the 50 Hz pure tone sinusoidal signal and white noise signal. But only the FXBPNN and the FEBPNN algorithms are effective on the harmonic noise signal caused by nonlinearity. Simulations of the two algorithms also suggest that the two algorithms have a similar performance. Considering the computational load shown in the previous section, the FEBPNN algorithm is a more practical method. Fig. 7 shows the mean square error in error microphone versus the number of iterations, from the result shown in Fig. 7, it can be seen that the converge rate of the FEBPNN is acceptable. Other simulations are followed to illustrate the performance of the FEBPNN.

Case 3: On the basis of case 2, the primary signal is changed after the system has entered into steady-state phase. Fig. 8 shows the mean square error in error microphone versus the number of iterations. When the number of iterations is equal to 3000, the primary signal is changed from a 50 Hz sinusoidal signal superimposed onto a white noise signal to a 75 Hz sinusoidal signal superimposed onto a white noise signal.

Case 4: To continue with this line of simulation, after the system has entered into steady-state phase, the secondary path is regulated by increasing a delay of 2 unit samples. Fig. 9 shows the

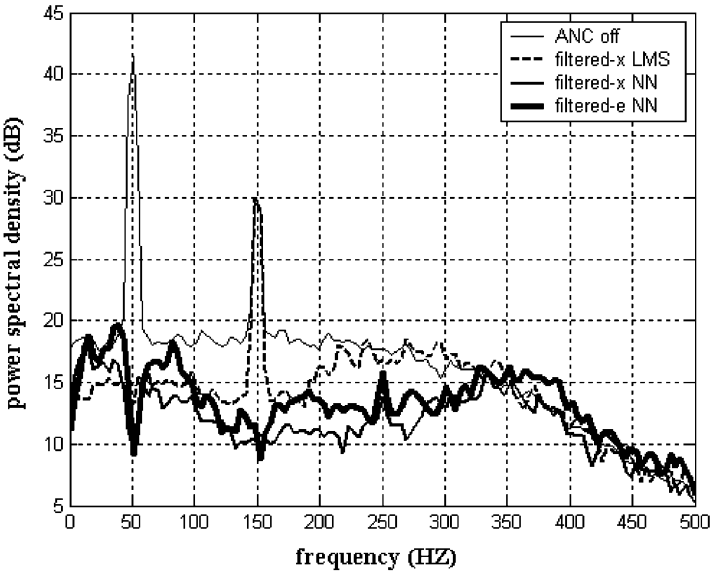

Fig. 6. Power spectrum of active noise canceling errors for nonlinear case.

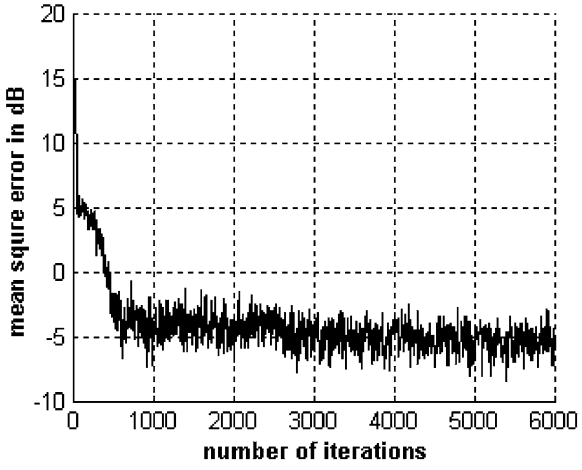

Fig. 7. The mean square error in error microphone versus the number of iterations.

mean square error in error microphone versus the number of iterations. When the number of iterations is equal to 2000, the secondary path is changed.

Case 5: On the basis of case 2, after the system has entered into steady-state phase, the primary path is regulated by letting $d(n) = -d(n)$. Fig. 10 shows the mean square error in error microphone versus the number of iterations. When the number of iterations is equal to 3000, the primary path is changed.

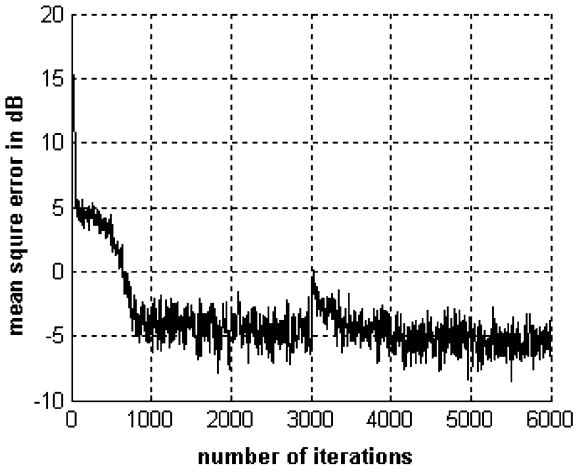

Fig. 8. The mean square error in error microphone versus the number of iterations when the primary signal is changed.

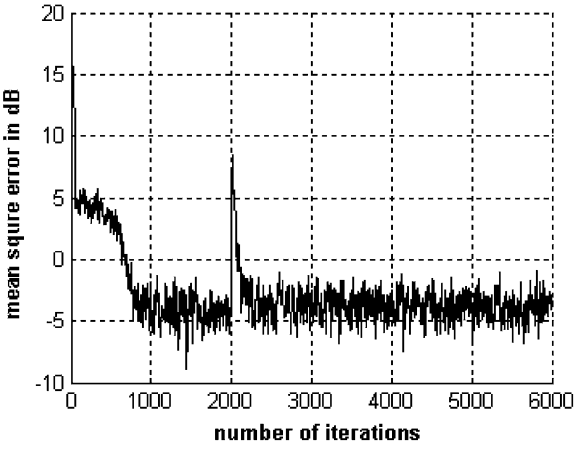

Fig. 9. The mean square error in error microphone versus the number of iterations when the second path is changed.

From the results shown in Figs. 8–10, it can be seen that the system has better tracking ability under changes of the primary signal, the primary path, the secondary path. Based on the simulation results, the FEBPNN algorithm is employed to reduce the harmonic noise signal caused by nonlinearity in the following experimental implementation.

4. Experimental implementation

4.1. Experimental setup

Now that the FEBPNN algorithm used in feedforward ANC system has been derived and verified by simulations, the next step is to verify the performance in an experimental setup. The

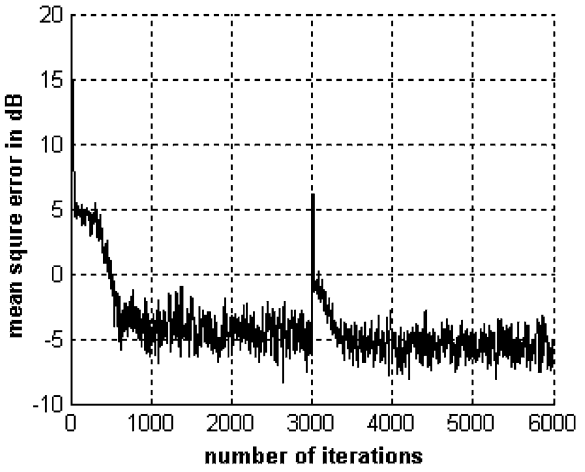

Fig. 10. The mean square error in error microphone versus the number of iterations when the primary path is changed.

test-bed chosen for this was a simple one-dimensional duct made of wood, with two loudspeakers whose resistance is 8 Ω and diameter is 16 cm. And two PZM86 microphones. The primary loudspeaker was used to generate the primary disturbance signal, and the canceling loudspeaker was used to generate the anti-noise signal. The reference microphone was used to measure the reference input signal, and the error microphone was used to measure the residual error signal. Fig. 11 shows the overall block diagram of the ANC system. Fig. 12A shows the photos of the duct used, Fig. 12B is the DSP platform in which the A/D converter AD7865 and the D/A converter AD7840 are included, and Fig. 12C is the signal processing board in which the filter MAX296, the amplifier MAX422 and the switched-mode power supply HAD2.5-5-N are included. The noise propagating down the duct was sampled by the upstream reference microphone and adaptively altered in the electronic feedforward path to produce the anti-noise signal to minimize the acoustic energy at the downstream error microphone. The length of the duct is 210 cm, and the cross-section is 20 cm × 18 cm, the distance between the reference microphone and the canceling loudspeaker is set to be 135 cm, such that the causality constraint is met [10]. The cutoff frequency of the duct for (1, 0) mode is about 858 Hz [11], in order to ensure that the acoustic wave propagates in the duct in the form of the plane-wave, the control bandwidth was limited to 800 Hz. The controller was implemented on a TI floating-point DSP TMS320VC33 and the TI C-language was used to implement the algorithm. The reference signal and the error signal measured by the corresponding microphone were first amplified by a amplifier MAX422, then filtered by a eighth-order Bessel low-pass filters MAX296 with cutoff frequency of 800 Hz and then converted from analog to digital by a AD7865, and finally processed by the DSP. The primary disturbance signal and the anti-noise signal outputted from DSP were first converted from digital to analog signal by AD7840, then smoothed by a eighth-order Bessel low-pass filters MAX296 with cutoff frequency of 800 Hz, then amplified by the HY5885B power amplifier, which was used to drive the corresponding loudspeakers. The gain level of the preamplifier could be fine-tuned to improve its performance.The sampling rate for the implementation was chosen to be 2 kHz, which satisfied the Nyquist criterion for sampling signals.

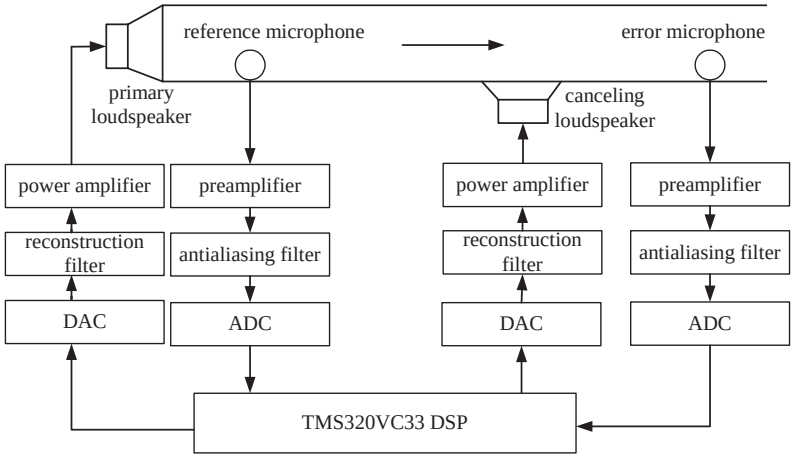

Fig. 11. The schematic diagram of ANC system.

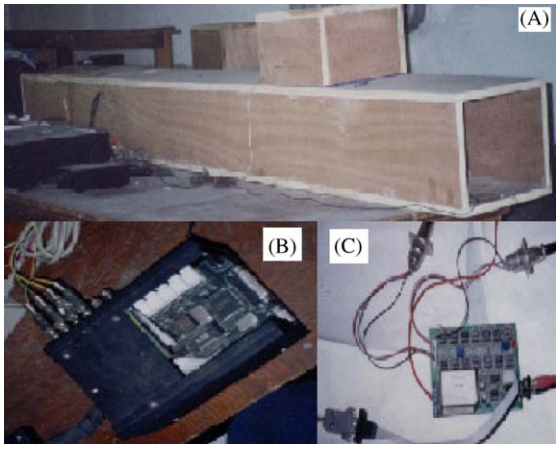

Fig. 12. The experimental duct system of active noise attenuation. (A) The duct, (B) the DSP platform, (C) signal processing board.

This experimental setup was selected both for convenience and simplicity. The aim of the experiments was purely to verify that the FEBPNN algorithm was capable of minimizing the harmonic noise signal caused by nonlinearity.

The frequency response characteristics of the duct are shown in Figs. 13 and 14, where Fig. 13 gives the frequency response of the primary path, and Fig. 14 gives the frequency response of the secondary path.

4.2. Secondary path identification

As mentioned above, the secondary path must be identified off-line during a training stage prior to the start of the ANC operation. The block diagram of the off-line secondary path modeling is

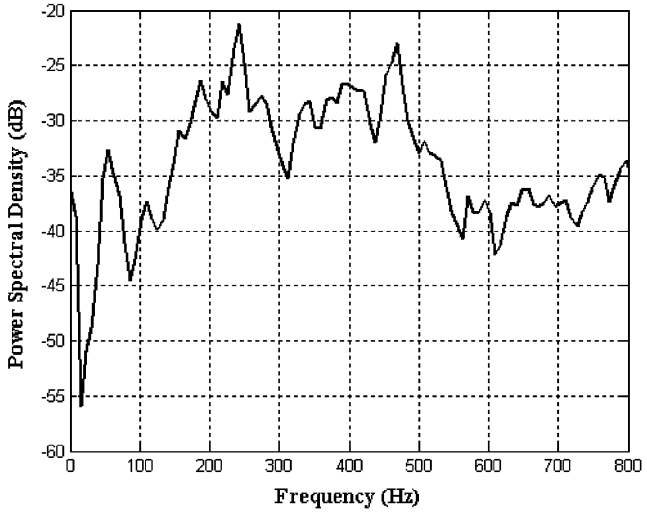

Fig. 13. Frequency response of primary path $P(Z)$.

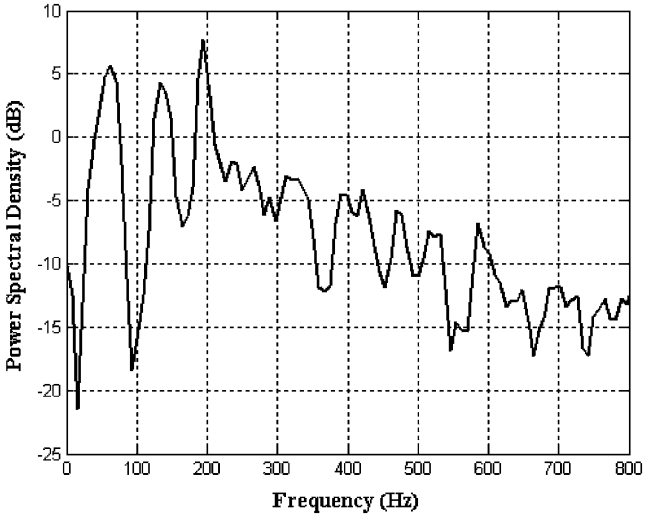

Fig. 14. Frequency response of secondary path $S(Z)$.

shown in Fig. 15. The secondary path $S(z)$ comprises the D/A converter, reconstruction filter, power amplifier, canceling loudspeaker, acoustic path from the canceling loudspeaker to error microphone, error microphone, anti-aliasing filter, and A/D converter. The secondary path transfer function was modeled as a 256-tap FIR filter, the entire off-line modeling took about 20 s. Fig. 16 gives the amplitude of the practical output of the system $e(n)$, the output of the adaptive filter $r(n)$ and the residual error signal $e'(n)$ for the last 200 sampling periods. Fig. 17 gives the power spectrum of the secondary path estimate, and the impulse response of the secondary path is shown in Fig. 18. From the results shown in Figs. 16 and 17, it can be seen that the transfer

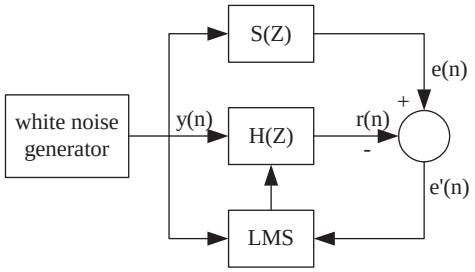

Fig. 15. The block diagram of the off-line secondary path identification.

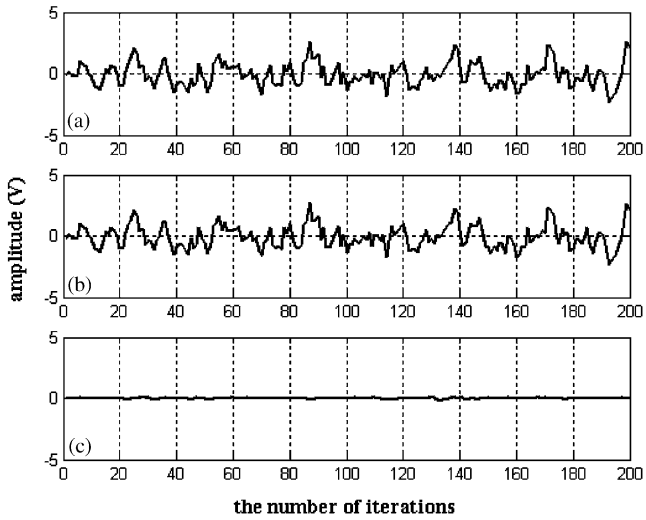

Fig. 16. Amplitude of secondary path output. (a) The amplitude of $e(n)$, (b) the amplitude of $r(n)$, (c) the amplitude of $e'(n)$.

function $H(z)$ is almost the same as the unknown transfer function $S(z)$. From the result shown in Fig. 18, it can be seen that the secondary path can be modeled using an FIR filter having 256 coefficients operating at a sampling rate of 2 kHz, and it is obvious that a delay of 5 unit samples exists in the secondary path.

4.3. Active noise cancelation

Once the secondary path has been identified exactly, the FEBPNN algorithm is used to attenuate the primary noise. The number of neurons used in the NN is 25-15-1. Two types of nodal output function were used. For the output node, the function was linear, providing the control signals with the capacity to vary over the positive/negative range required for control. For the hidden node, a sigmoidal nonlinear output function was used:

$$f(x) = \frac{1 - e^{-x}}{1 + e^{-x}}. \tag{34}$$

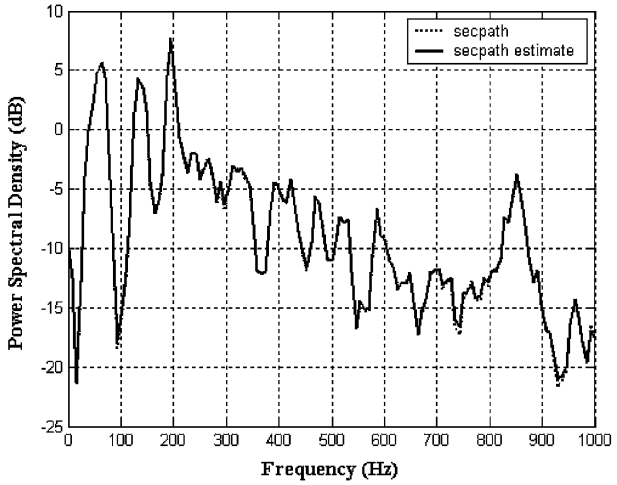

Fig. 17. Power spectrum of secondary path $S(Z)$.

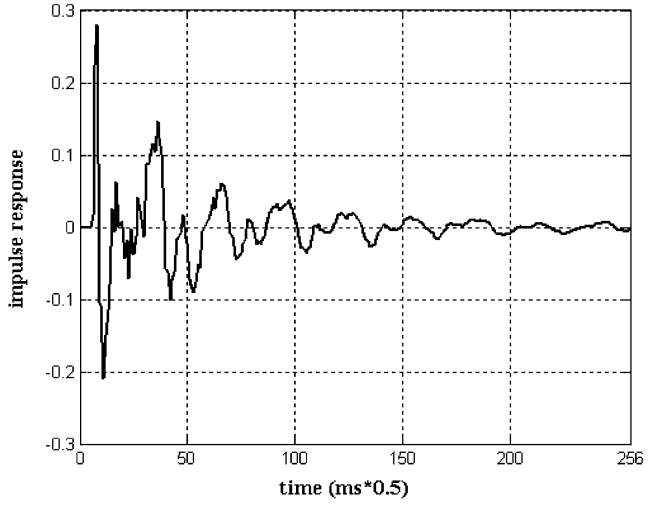

Fig. 18. Impulse response of secondary path $S(Z)$.

For comparative purposes, a 32-tap FIR filter controller adapted using the FXLMS algorithm was also implemented. The variable step size $\mu(n)$ was used for both the FEBPNN algorithm and the FXLMS algorithm, and was given by the following equation [2]:

$$\mu(n) = \frac{\alpha}{P_x(L + \Delta)}, \tag{35}$$

where $0 < \alpha < 1$, P_x is the normalized power of the input signal. For the FEBPNN algorithm, L is the length of the tap-delay line in the reference input signal, whereas, for the FXLMS algorithm, L is the order of the filter. Δ is the number of samples corresponding to the overall delay in the secondary path. In this experiment, α was selected as 0.0198 and 0.0118 for the FEBPNN and the

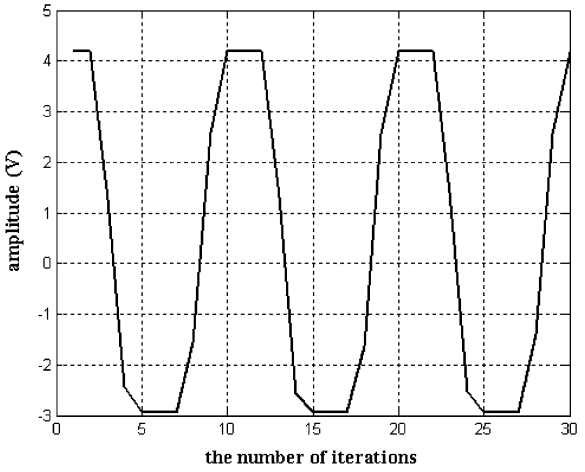

Fig. 19. The output signal of the error microphone.

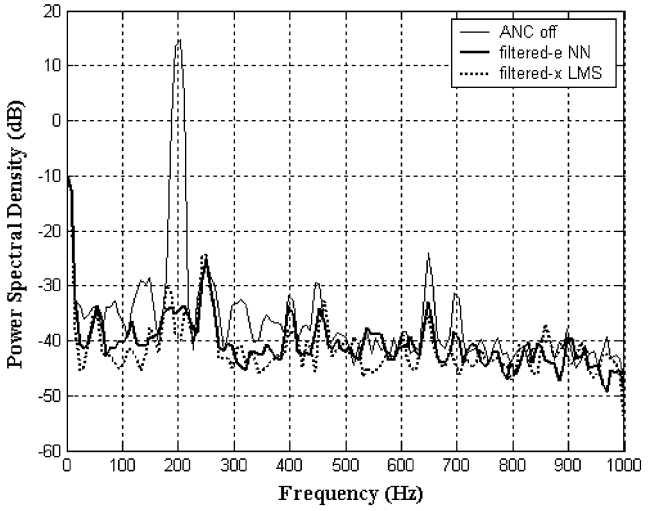

Fig. 20. Error signal spectrum for linear case.

FXLMS algorithm, respectively. It was found from the experiments that the system itself exhibits nonlinear characteristics when the amplitude of the primary noise is increased to a certain degree, so the nonlinearity was introduced into the experimental arrangement by increasing the amplitude of the primary noise, to further strengthen the relative intensity of the nonlinearity, the preamplifier of the error microphone was adjusted to ensure that it is driven to saturation. Fig. 19 shows the error microphone output signal when a 200 Hz pure tone sinusoidal signal is used as the input noise signal.

Case 1: The first case to consider is a simple linear one, the error signal spectrum during primary excitation and under control with a 200 Hz pure tone sinusoidal signal disturbance is shown in Fig. 20. From the results shown in Fig. 20, it can be seen that the control achieved by using the NN controller looks identical to the linear control case.

Although the NN algorithm is mainly used in nonlinear systems, it is important that how effective it works on linear case for two reasons: first, it demonstrates that the architecture/algorithm combination works, at least for a linear case. Second, if the NN is used to supplant linear filters in adaptive feedforward active control systems, it must be able to handle linear control problems effectively [4].

Case 2: The second case investigates the nonlinear performance of the linear and FEBPNN-based controller. The nonlinearity was introduced into the experimental arrangement by increasing the amplitude of the primary noise, and the preamplifier of the error microphone was driven to saturation. The resultant error signal spectrum for nonlinear case is shown in Fig. 21. From the results shown in Fig. 21, it can be seen that the NN controller performance is vastly superior to that of the linear controller. Not only was the primary tone level reduced, but also the harmonic noise in the spectrum was reduced. Fig. 22 shows the error signal in error microphone versus the number of iterations, from the result shown in Fig. 22, it can be seen that the convergence rate of the FEBPNN is acceptable.

Case 3: The third case investigates the tracking ability of the FEBPNN-based controller. The primary signal was changed after the system had entered into steady-state phase. Fig. 23 shows the error signal in error microphone versus the number of iterations. Due to being limited by the memory space of the DSP, it is not possible to record the entire change procedure of the error signal, so in Fig. 23, the number of iterations is not continuous, the curve between 0 and 100 iterations shows the error signal when the ANC system turns off, the curve between 25 000 and 25 750 iterations shows the error signal when the ANC system turns on and has entered into steady-state phase, at this time, the disturbance signal is a 200 Hz sinusoidal signal. When the number of iterations is equal to 25 750, the disturbance signal was changed from a 200 Hz sinusoidal signal to a 55 Hz sinusoidal signal, the curve between 25 750 and 26 000 iterations shows the error signal during first 250 iterations after the disturbance signal was changed, the

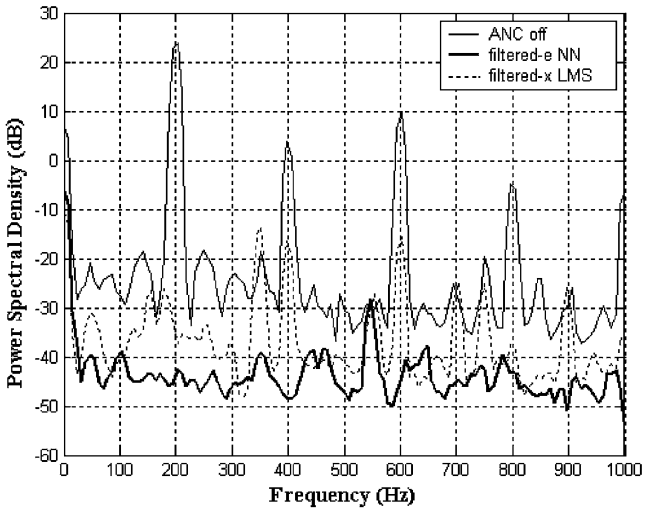

Fig. 21. Error signal spectrum for nonlinear case.

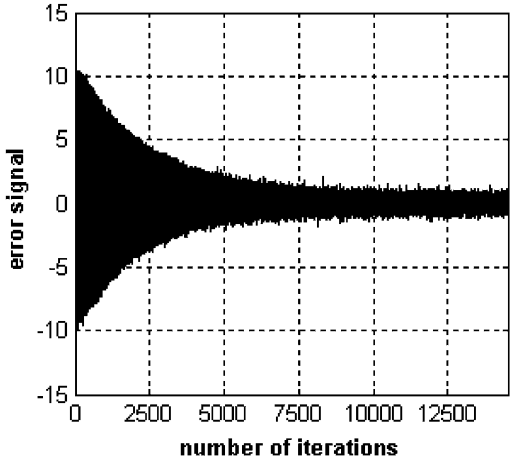

Fig. 22. Error signal versus number of iterations.

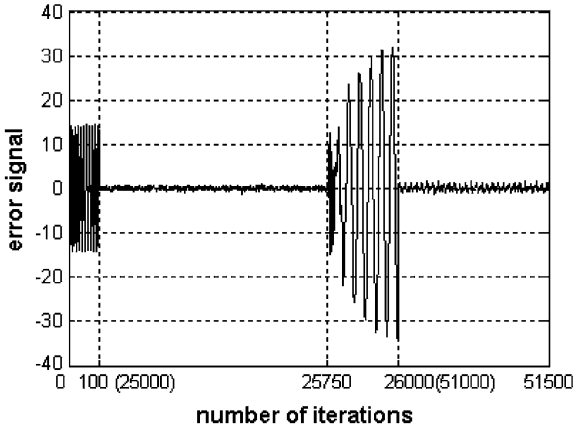

Fig. 23. Error signal versus number of iterations when the primary signal is changed.

curve between 51 000 and 51 500 iterations shows the error signal after the system had entered into steady-state phase again. From the results shown in Fig. 23, it can be seen that the system has better robust performance under change of the primary signal.

Case 4: As discussed in previous sections, the secondary path is estimated off-line prior to the operation of the ANC system. For some applications, the secondary path may be time varying, and it is desirable to estimate the secondary path on-line when the ANC system is in operation. So, the fourth case investigates the self-adaptive ability of the FEBPNN-based controller when the secondary path is time varying.

An on-line secondary path modeling technique using additive random noise is illustrated in Fig. 24 [2], a zero-mean white noise $v(n)$ is internally generated and is added to the secondary signal $u(n)$ to drive the secondary source. The adaptive filter $H(Z)$ using LMS algorithm is

connected in parallel with the secondary path $S(Z)$, however, the input signal used for $H(Z)$ is the random noise $v(n)$ only.

The FEBPNN algorithm with an on-line secondary path modeling (OSPM) using additive random noise is as follows:

Step (1): Input the reference signal $X(n)$ and the error signal $e(n)$.

Step (2): Calculate the output of the FEBPNN-based controller $u(n)$.

Step (3): Calculate the summation of the output of the FEBPNN-based controller $u(n)$ and the additive random noise $v(n)$.

Step (4): Output the signal $u(n) + v(n)$ to drive the secondary source.

$$\begin{aligned} h_i(n+1) &= h_i(n) + \mu f(n) v(n) \\ &= h_i(n) + \mu v(n)[d(n) + y(n) + c(n) - r(n)] \\ &= h_i(n) + \mu v(n)[c(n) - r(n)] + \mu v(n) b(n), \end{aligned} \tag{36}$$

where $b(n) = d(n) + y(n) = d(n) + u(n)s(n)$, indicate the component of the error due to the original noise .

Step (5): Update the weights of the adaptive filter $H(Z)$ using LMS algorithm.

Step (6): Update the weights in the output layer of the FEBPNN-based controller using Eq. (30).

Step (7): Update the weights in the hidden layer of the FEBPNN-based controller using Eq. (31).

Step (8): Repeat the procedure for the next iteration.

Fig. 25 shows the error signal in error microphone versus the number of iterations. Similar to case 3, due to being limited by the memory space of the DSP, the number of iterations is not continuous; the curve between 0 and 100 iterations shows the error signal when the ANC system turns off, the curve between 1 80 000 and 1 80 500 iterations shows the error signal when the ANC system turns on and has entered into steady-state phase, at this time, the distance between the

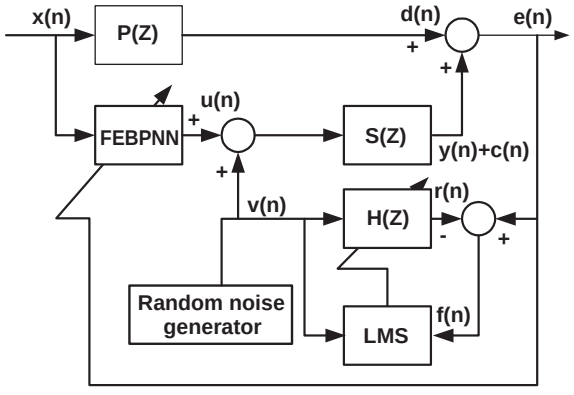

Fig. 24. Block diagram of ANC system based on FEBPNN with on-line secondary path modeling using additive random noise.

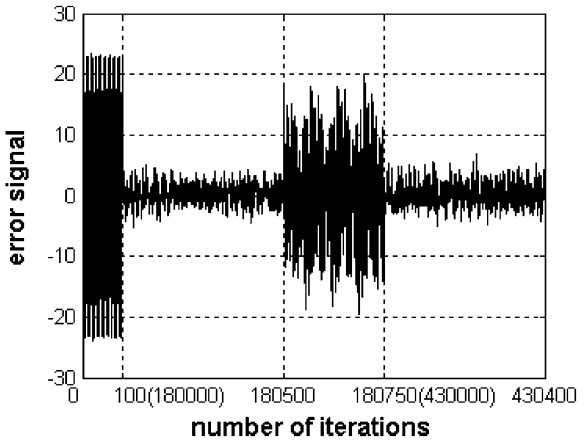

Fig. 25. Error signal versus number of iterations when the secondary path is changed.

error microphone and the canceling loudspeaker is set to be 50 cm. When the number of iterations is equal to 1 80 500, move the error microphone such that the distance between the error microphone and the canceling loudspeaker is 10 cm, the curve between 1 80 500 and 1 80 750 iterations shows the error signal during first 250 iterations after the secondary path was changed, the curve between 4 30 000 and 4 30 400 iterations shows the error signal after the system has entered into steady-state phase again. From the results shown in Fig. 25, it can be seen that the system can resist disturbance and has better robust performance under change of the secondary path. However compared to the results using the off-line secondary path estimation, it was found that the convergence rate of the FEBPNN with OSPM is slower than that of the FEBPNN with off-line secondary path modeling, the reason is that for on-line modeling, $b(n)$ acts like an uncorrelated plant noise, after convergence, this residual noise due to $b(n)$ will perturb the adaptive weights of $H(Z)$, resulting in a misalignment. In most ANC applications, the interference $b(n)$ is much larger than the excitation signal $v(n)$ and the convergence rate of filter $H(Z)$ is therefore very slow. In other words, it is obvious that the undesired term $\mu\, v(n)b(n)$ in Eq. (36) is a disturbance that is frustrating convergence of the LMS algorithm and therefore degrade the performance of the adaptive filter $H(Z)$.

5. Discussion of results

The experimental results presented in the previous section clearly demonstrated the ability of the NN based on the FEBPNN algorithm to provide significant levels of noise attenuation in a feedforward active control scheme. From the experimental results shown in Figs. 20 and 21, attenuation of the FEBPNN algorithm was seen to be equal in performance to the linear control scheme for a linear control problem, and to be superior to the linear control scheme when the control problem has some inherent nonlinearity.

Although the NN based on the FEBPNN algorithm has aforementioned advantages, there are many theoretical and practical problems needed to be resolved. First of all, the principle problem

is a lack of predictability and consistency of the NN. For a given set of conditions, and due to the local minimum in the error criterion, the levels of noise control obtained may not be consistent. The random initial weights were not sufficient to enable the algorithm to avoid getting trapped in it. Next, in spite of the universal approximation theorem which states that a continuous feedforward NN with a single hidden layer can approximate nonlinear mappings arbitrarily well [12,13], many theoretical issues are still not fully resolved for multilayer NN: for example, the number of neurons needed in a single hidden layer to achieve a given accuracy of approximation to a nonlinear function, the number of layers needed to give the most efficient approximation to a nonlinear function, or how the weights in these neurons should be optimally determined [5]. Therefore, based on the experimental work conducted and presented in this paper, the following can be said: the NN based on the FEBPNN algorithm for feedforward ANC shows the potential to equal in performance to the linear control scheme for a linear control problem and to have superior performance for nonlinear problems.

6. Conclusions

The NN controller based on the FEBPNN algorithm has been developed for use in feedforward ANC schemes. Some simulations were presented to compare the FXBPNN algorithm, the FEBPNN algorithm, and the FXLMS algorithm. Following this, the NN controller based on the FEBPNN algorithm was implemented on a Texas Instruments DSP TMS320VC33. For comparison, a FIR filter controller adapted using the FXLMS algorithm was also implemented on the same DSP platform, two approaches of implementing the controller are verified experimentally with a model of a duct system. The simulations and the experimental verification tests indicated that the NN controller was equal in performance for a linear control problem and superior for a nonlinear one compared to a linear control scheme. For this to be fully realized, however, a great deal of work needs to be done. Efforts are being carried out to improve on the predictability and consistency in using the nonlinear NN controller, and these approaches will be presented in future papers.

Acknowledgements

This research is supported by Science Funds of Educational Committee of Beijing City (KM200311232137) and Training Funds for Elitist of Beijing.

References

[1] S.J. Elliott, P.A. Nelson, Active noise control, *IEEE Signal Processing Magazine* 10 (1993) 12–35.
[2] S.M. Kuo, D.R. Morgan, Active noise control: a tutorial review, *IEEE Proceedings* 87 (1999) 973–993.
[3] C.C. Ku, K.Y. Lee, Diagonal recurrent neural networks for dynamic systems control, *IEEE Transactions on Neural Networks* 6 (1995) 144–156.

[4] S.D. Snyder, Nobuo Tanaka, Active control of vibration using a neural network, *IEEE Transactions on Neural Networks* 6 (1995) 819–828.
[5] S.J. Elliott, *Signal Processing for Active Control*, Academic Press, London, 2001.
[6] R. Jha, J. Rower, Experimental investigation of active vibration control using neural networks and piezoelectric actuators, *Smart Materials & Structures* 11 (2002) 115–121.
[7] S.M. Kuo, D. Vijayan, Adaptive algorithms and experimental verification of feedback active noise control systems, *Noise Control Engineering Journal* 42 (1994) 37–46.
[8] S.M. Kuo, D.R. Morgan, *Active Noise Control Systems—Algorithms and DSP Implementations*, Wiley, New York, 1996.
[9] Q.Z. Zhang, Y.L. Jia, Active noise hybrid feedforward/feedback control using neural network compensation, *Journal of Vibration and Acoustics* 124 (2002) 100–104.
[10] P. Minogue, N. Rankin, J. Ryan, Short duct server fan noise cancellation, *ACTIVE1999*, Florida, USA, 1999, pp. 539–550.
[11] Jwusheng Hu, Jyh-Feng Lin, Feedforward active noise controller design in ducts without independent noise source measurements, *IEEE Transactions on Control System & Technology* 8 (2000) 443–455.
[12] G. Cybenko, Approximation by superposition of a sigmoidal function, *Mathematics of Control, Signals, and Systems* 2 (1989) 303–314.
[13] K. Funahashi, On the approximate realization of continuous mappings by neural networks, *Neural Networks* 2 (1989) 183–192.