

Twelfth International Congress  
on Sound and Vibration

## NONLINEAR ACTIVE NOISE CONTROL OF HARMONIC SOUND BY MEANS OF NEURAL NETWORKS

André Jakob, Michael Möser

Institut für Technische Akustik, Technische Universität Berlin  
Sekt. TA7, Einsteinufer 25, D-10587 Berlin, Germany  
andre.jakob@tu-berlin.de

### Abstract

Some active noise and vibration control applications demand for secondary sources which show nonlinear behaviour. An example is the so-called electro pneumatic loudspeaker. This device generates low frequency sound by means of a modulated compressed air-flow. This is similar to the principle of sirens but differs in such a way that a plate with holes (the slider) is not rotating but moving back and forth driven by a shaker, for example. In principle the input to the sound source can be any waveform rather than the "fixed waveform" of the rotating siren which can be only controlled by the rotational speed. The mechanism of sound generation itself as well as the driving mechanism, the latter for example due to friction of the slider, is nonlinear. Other applications might contain piezo-electric secondary sources or other nonlinear devices. When a nonlinear secondary source is used in active noise control applications a nonlinear controller should be used. Amongst other things, one possibility is the application of neural networks. In the presented work a neural network is used to model the nonlinear secondary source. As a feedforward controller either another neural network or a (linear) FIR filter is used. The latter might be reasonable for control of harmonic sound. The derivation of the adaptation algorithm will be outlined briefly and simulation results will be given and discussed.

### INTRODUCTION

The neural networks used here are based on the so called *multi layer perceptron* [1]. When the inputs of the neural network are delayed versions of a sampled

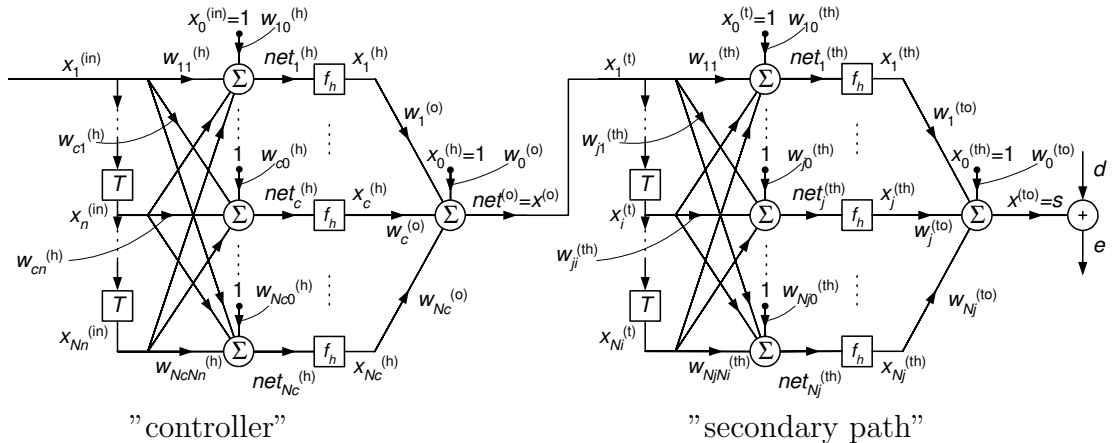


Figure 1: *Feedforward control scheme with neural network controller and neural network secondary path model.*

signal, i.e. the stored values of a delay-chain, one obtains a so called *time-delay neural network* (TDNN).

It is assumed here throughout the text that the nonlinear secondary path is modelled prior to active control as such a TDNN. Procedures for neural network based nonlinear system identification are given in [2]. From linear system identification theory a lot of different model structures are known, e.g. FIR, ARX, ARMAX, OE, etc. It is possible to define similar model structures for (nonlinear) neural network models [2]. The simplest is the NNFIR model and this is used here. It has the structure given in the right hand parts of Figures 1 and 2.

If a neural network is also used as the controller the structure in Figure 1 results. When thinking of harmonic noise to be controlled this might motivate for the use of a (linear) FIR filter based controller. The idea is to pre-generate all the possibly excited harmonics and simply shift them in phase and magnitude by a FIR filter. This would greatly reduce the computational effort. Figure 2 shows the structure with the FIR filter controller and the neural network secondary path model. The derivation of the adaptation algorithm is based on that given in [3, 4]. It is an extension of the well-known back-propagation algorithm for the filtered-x case. It can be shown that the filtered-x LMS algorithm is a special case of the more generic *"filtered-x back-propagation algorithm"*.

## DERIVATION OF THE ADAPTATION/TRAINING ALGORITHM

Here is shown the derivation of the adaptation algorithm for a neural network controller and the derivation of the adaptation algorithm for a FIR filter controller for the use in active noise control applications. The derivation shown here is restricted to the SISO case. The derivation for the MIMO case is shown in [3, 4].

The structure of the neural networks used here is the multilayer perceptron (MLP) with one hidden layer with nonlinear activation functions and one output neuron with a linear activation function. The well-known back-propagation algorithm [1] is used to derive the adaptation of the controller weights. It is assumed

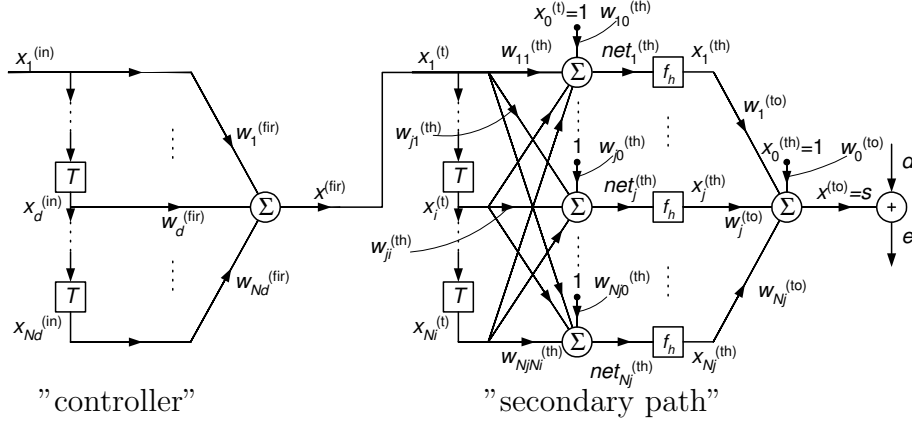


Figure 2: *Feedforward control scheme with FIR filter controller and neural network secondary path model.*

here that the neural network secondary path model identified prior to control is fixed during the adaptation of the controller weights.

### The gradient method

The goal of the adaptation of the controller weights is to minimize the instantaneous squared error signal  $e^2(k)$ , where  $k$  is used here for the discrete time-step variable. In the back-propagation algorithm, which is a gradient method, the weights of the network are updated with the negativ gradient of this (instantaneous) squared error. The generic update rule thus is

$$w(k+1) = w(k) - \mu \Delta w(k). \quad (1)$$

The error signal  $e(k)$  in active noise control applications is the sum of the primary disturbance signal  $d(k)$  measured at the error sensor and the secondary signal  $s(k)$  coming from the secondary source(s) measured at this sensor, i.e.

$$e(k) = d(k) + s(k). \quad (2)$$

Thus, the gradient becomes

$$\Delta w(k) = \frac{\partial e^2(k)}{\partial w(k)} = 2e(k) \frac{\partial e(k)}{\partial w(k)} = 2e(k) \frac{\partial s(k)}{\partial w(k)}. \quad (3)$$

### The equation for the secondary path model

The task now is to express the signal  $s(k)$  such that the partial derivatives with respect to the controller weights can be derived. First, the equation for the secondary path model will be set up directly from Figure 1. Some definitions must be detailed here (cf. Figure 1). The weights of the single output layer node are

denoted as  $w_j^{(to)}$  numbered as  $j = 0, 1, \dots, N_j$ . The index  $j = 0$  belongs to the weight  $w_0^{(to)}$ , which is the weight of the bias input to the output node. The activation functions in the hidden layer of the neural network secondary path model are denoted by  $f_{th}(\cdot)$ . Identical activation functions are assumed for all nodes in the hidden layer of the neural network secondary path model. The weights in the hidden layer are denoted with  $w_{ji}^{(th)}$ , which is the weight from the  $i$ -th input node to the  $j$ -th hidden layer node. The index  $i = 0$  belongs to the weights  $w_{j0}^{(th)}$ , which are the weights of the bias inputs to the hidden layer nodes. The input layer consists of a time-delay chain, because the neural network shall model a dynamic system, with  $N_i - 1$  delays resulting in a  $N_i$ -stage input layer. The (single) input to the secondary path model is  $x^{(t)}(k)$ . With this definitions the governing equation for the neural network secondary path model can be stated directly from Figure 1 as

$$s(k) = w_0^{(to)} + \sum_{j=1}^{N_j} w_j^{(to)} f_{th} \left( w_{j0}^{(th)} + \sum_{i=1}^{N_i} w_{ji}^{(th)} x^{(t)}(k - i + 1) \right). \quad (4)$$

### The training algorithm for the neural network controller

In the same way the equation for the neural network controller can be stated from Figure 1. The output neuron weights are denoted as  $w_c^{(o)}$  with  $c = 0, 1, \dots, N_c$ , the activation functions are denoted as  $f_h(\cdot)$ , the hidden layer weights are denoted as  $w_{cn}^{(h)}$  with  $n = 0, 1, \dots, N_n$ , and the controller input delay chain provides the neural network with  $N_n$  samples of the reference signal  $x^{(in)}(k)$ . The equation for the controller neural network therefore is

$$x^{(t)}(k) = w_0^{(o)}(k) + \sum_{c=1}^{N_c} w_c^{(o)}(k) f_h \left( w_{c0}^{(h)}(k) + \sum_{n=1}^{N_n} w_{cn}^{(h)}(k) x^{(in)}(k - n + 1) \right). \quad (5)$$

### *Adaptation of the output layer weights*

The philosophy of the back-propagation algorithm is to adapt the weights backwards starting from the output layer to the eventually multiple hidden layers. This is not really necessary in this simple case considered here, but to be conform with the derivation of the standard back-propagation algorithm it might be useful to start with the adaptation of the controller neural network output layer weights. For building up the derivative in Equation (3) it is useful to re-express Equation (5) with the output signals  $x_c^{(h)}(k)$  of the activation functions  $f_h(\cdot)$ , i.e.

$$x^{(t)}(k) = w_0^{(o)}(k) + \sum_{c=1}^{N_c} w_c^{(o)}(k) x_c^{(h)}(k). \quad (6)$$

Inserting Equation (6) into Equation (4) and building up the gradient according to Equation (3) one ends up with

$$\begin{aligned}\Delta w_c^{(o)}(k) &= 2e(k) \frac{\partial s(k)}{\partial w_c^{(o)}} \quad (7) \\ &= \begin{cases} 2e(k) \sum_{j=1}^{N_j} w_j^{(to)} f'_{th} \left( net_j^{(th)}(k) \right) \sum_{i=1}^{N_i} w_{ji}^{(th)} x_c^{(h)}(k-i+1) & \text{if } c = 1, \dots, N_c \\ 2e(k) \sum_{j=1}^{N_j} w_j^{(to)} f'_{th} \left( net_j^{(th)}(k) \right) \sum_{i=1}^{N_i} w_{ji}^{(th)} & \text{if } c = 0 \end{cases} \quad (8)\end{aligned}$$

wherein  $f'_{th}(\cdot)$  is the (outer) derivative of  $f_{th}(\cdot)$ . For convenience the instantaneous values of the input signals to the activation functions are abbreviated as  $net_j^{(th)}(k)$ .

### ***Adaptation of the hidden layer weights***

In the same way the gradient with respect to the controller hidden layer weights can be obtained. Insertion of Equation (5) into Equation (4) directly yields

$$s(k) = w_0^{(to)} + \sum_{j=1}^{N_j} w_j^{(to)} f_{th} \left( w_{j0}^{(th)} + \sum_{i=1}^{N_i} w_{ji}^{(th)} \left( w_0^{(o)} + \sum_{c=1}^{N_c} w_c^{(o)} f_h \left( w_{c0}^{(h)} + \sum_{n=1}^{N_n} w_{cn}^{(h)} x_n^{(in)}(k-i+1) \right) \right) \right) \quad (9)$$

and the gradient becomes

$$\begin{aligned}\Delta w_{cn}^{(h)}(k) &= 2e(k) \frac{\partial s(k)}{\partial w_{cn}^{(h)}} \\ &= \begin{cases} 2e(k) \sum_{j=1}^{N_j} w_j^{(to)} f'_{th} \left( net_j^{(th)}(k) \right) \sum_{i=1}^{N_i} w_{ji}^{(th)} w_c^{(o)}(k-i+1) f'_h \left( net_c^{(h)}(k-i+1) \right) x_n^{(in)}(k-i+1) & n = 1, \dots, N_n \\ 2e(k) \sum_{j=1}^{N_j} w_j^{(to)} f'_{th} \left( net_j^{(th)}(k) \right) \sum_{i=1}^{N_i} w_{ji}^{(th)} w_c^{(o)}(k-i+1) f'_h \left( net_c^{(h)}(k-i+1) \right) & n = 0 \end{cases} \quad (10)\end{aligned}$$

wherein  $f'_h(\cdot)$  is the (outer) derivative of  $f_h(\cdot)$ . Again, for convenience the instantaneous values of the input signals to the activation functions are abbreviated as  $net_c^{(h)}(k-i+1)$ , now for the delayed versions of the controller activation function input signals. Note, that the input signals  $x_n^{(in)}(k-i+1)$  to the controller neural network can also be written in the form  $x^{(in)}(k-i+1-n+1)$  because they are also outputs of a delay chain. But here it might be a little bit more convenient to use the form given in Equation (10).

### **The training algorithm for the FIR filter controller**

The idea behind using a FIR filter based controller instead of a neural network based controller is as follow. Suppose an active noise control task for harmonic sound only, no random sound. Suppose further, that all frequencies are available in the reference signal, i.e. no additional frequencies are excited due to the nonlinearities in the actuator, only harmonic components. Then, all these known

frequencies simply have to be shifted in phase and amplitude like in conventional (linear) active noise control applications. However, the adaptation of the filter coefficients must be done in such a way that the nonlinearities of the secondary path are taken into account.

The main advantage of the FIR filter controller instead of the neural network controller would be the great reduction in processing power since only a FIR filter must be executed and only the gradient for one layer has to be evaluated. Moreover, no nonlinear activation functions inside the controller have to be computed.

It is of course now an absolutely trivial task to derive the adaptation for the FIR filter controller. From Figure 2 it can be seen, that the FIR filter is simply a rather extreme simplification of a neural network, i.e. a 'neural network' with no hidden layer and only one linear output neuron in the output layer without any bias weights. Thus one can simply use the gradient just derived in the previous section for the output layer weights omitting the bias weight. Some slightly new denotations are introduced here to distinguish the gradients for the two controllers. From Figure 2 can be deduced that the equation describing the FIR filter controller with  $N_d$  filter coefficients/weights is

$$x^{(t)}(k) = \sum_{d=1}^{N_d} w_d^{(fir)} x_d^{(in)}(k) \quad (11)$$

with  $x_d^{(in)}(k) = x^{(in)}(k - d + 1)$ . Thus, completely analogous to Equation (7) and (8) the gradients for the update of the FIR filter weights become

$$\begin{aligned} \Delta w_d^{(fir)}(k) &= 2e(k) \frac{\partial s(k)}{\partial w_d^{(fir)}} \\ &= 2e(k) \sum_{j=1}^{N_j} w_j^{(to)} f'_{th} \left( net_j^{(th)}(k) \right) \sum_{i=1}^{N_i} w_{ji}^{(th)} x_d^{(in)}(k - i + 1) \quad , d = 1, \dots, N_d. \end{aligned} \quad (12)$$

## SIMULATION RESULTS

Some simulation were performed using a rather simple neural network secondary path model consisting of 5 input taps and 5 hidden layer neurons, which was computed from experimental data as a very rough model of the electro-pneumatic sound source. The disturbance was a mono-frequent signal with fixed frequency at 4.5% of the sampling frequency. Simulations with the neural network controller as well as simulations with the FIR filter controller were performed.

Figure 3 shows spectrograms of the error signal. Note, that for better visibility the frequency axis is always reversed. While the first second the controllers were switched off, thus all plots show the same mono-frequent disturbance at the beginning.

Figure 3a shows the results obtained using a neural network controller with 5 input taps and 10 hidden layer units. It shows up that the disturbance signal

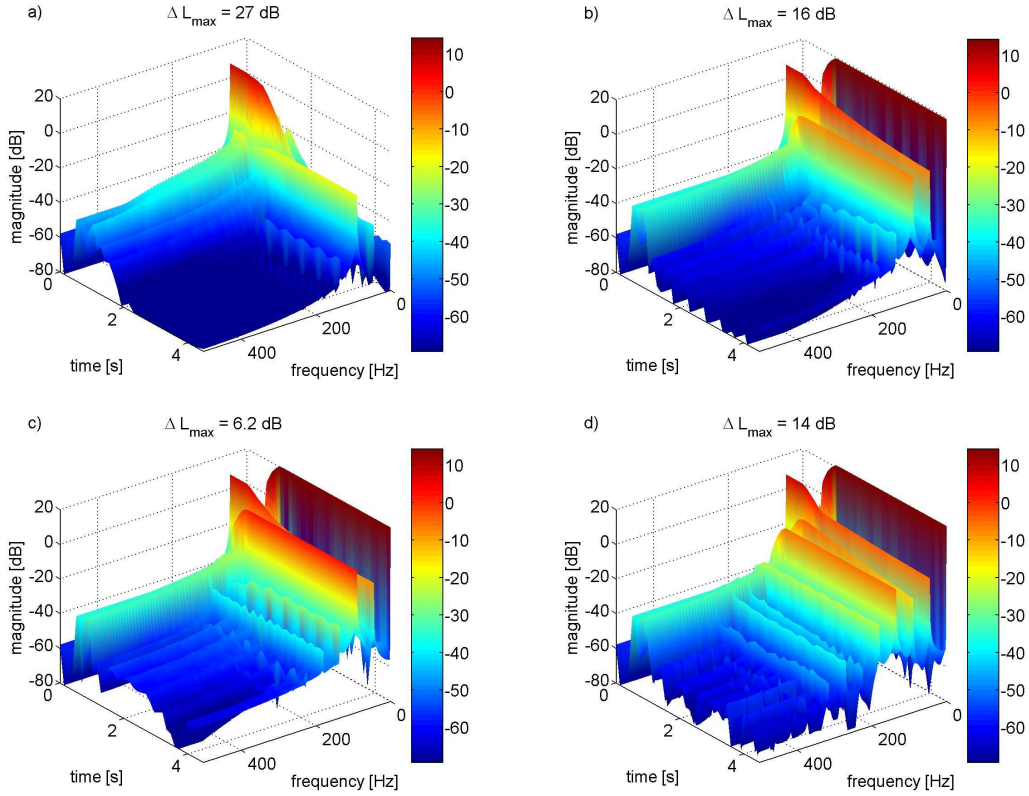


Figure 3: Simulation results: a) neural network controller with single frequency reference signal, b) FIR controller with single frequency reference signal and  $\mu = 0.2$ , c) FIR controller with single frequency reference signal and  $\mu = 0.5$ , d) FIR controller with reference signal containing two frequencies and  $\mu = 0.2$ .

is reduced greatly by more than 30 dB, but the overall success is limited by the first harmonic of the disturbance which is excited due to the nonlinearities in the secondary path. Nevertheless, the difference between the level of the error signal without control and the highest frequency component of the error signal with control was 27 dB at the end of the simulation. The result differs somewhat for different (random) initializations of the neural network controller.

Figures 3b to d show the results obtained with a FIR filter controller. In Figure 3b a FIR filter controller with 2 filter coefficients was used, because this number is sufficiently to arbitrarily influence a pure sine signal in phase and magnitude. The obtained reduction is much lower, because the (linear) FIR filter is not able to produce harmonics of the reference signal, which the controller must use to compensate for the harmonics the secondary path introduces in the error signal. The result is strongly dependent on the step size  $\mu$  of the adaptation algorithm. In Figure 3b was chosen a nearly optimal  $\mu$ , whereas in Figure 3c it was used a improper  $\mu$  and the result was worse.

It was the motivation for the use of a (linear) FIR filter, that a pre-generation

of harmonics could be used in combination with the FIR filter to overcome the nonlinear disturbances of the secondary path. For the simulation of Figure 3d additionally the first harmonic of the disturbance signal was used in the reference signal — and of course a FIR filter with 4 filter coefficients. It showed up in this case, that in spite of nearly optimal adjustment of  $\mu$  no further improvements could be achieved. Rather, even more harmonics are excited in the secondary path. An increase of this effect was observed, when introducing more harmonics.

Additionally, the spectra of the simulations with the FIR filter controllers all contain a DC component due to the fact, that the secondary path introduces this component and the FIR filter controller is not able to compensate for that. This is in contrast to the neural network controller, which is able to compensate for it.

### SUMMARY AND CONCLUDING REMARKS

The derivation of adaptation algorithms for a neural network controller and a FIR filter controller for use in active noise control systems which contain a nonlinear secondary path was shown. It was assumed in all cases that the nonlinear secondary path is identified prior to active control as a neural network model. Simulations were performed showing the feasibility of the algorithms to reduce tonal noise while presence of a nonlinear secondary path, which introduces unwanted harmonics into the system. It showed up, that the neural network controller performs much better than the FIR filter controller even when pre-generated harmonics are used for the FIR filter controller.

Of course, deeper investigations have to be done, e.g. with multi-frequent disturbance signals or larger secondary path models. Additionally tests will be done with nonlinear secondary path models containing no offset, i.e. no DC component, which is normally the case in acoustic actuators. Because the success with the FIR filter controller strongly depends on the step size an automatic step size adaptation will be investigated for the algorithms, too.

### ACKNOWLEDGEMENTS

This work was sponsored by the DFG, Deutsche Forschungsgemeinschaft, project name: "Nichtlineare Antischallquellen".

### REFERENCES

- [1] S. Haykin. *Neural Networks, A Comprehensive Foundation*. 2nd ed. (Prentice-Hall Inc. 1999)
- [2] M. Norgaard, O. Ravn, N.K. Poulsen, L.K. Hansen. *Neural Networks for Modelling and Control of Dynamic Systems*. (Springer-Verlag London Limited 2000)
- [3] C.H. Hansen, S.D. Snyder. *Active Control of Noise and Vibration*. (Chapman and Hall 1997)
- [4] S.D. Snyder, N. Tanaka. Active noise control using a neural network, *IEEE Transactions on Neural Networks*, 6(4):819–828, 1995