Välimäki, Vesa; Rämö, Jussi

Neurally Controlled Graphic Equalizer

Published in:
IEEE/ACM Transactions on Audio, Speech, and Language Processing

DOI:
10.1109/TASLP.2019.2935809

Published: 01/12/2019

Document Version
Publisher's PDF, also known as Version of record

Neurally Controlled Graphic Equalizer

Vesa Välimäki, *Fellow, IEEE*, and Jussi Rämö

***Abstract*—This paper describes a neural network based method to simplify the design of a graphic equalizer without sacrificing the accuracy of approximation. The key idea is to train a neural network to predict the mapping from target gains to the optimized band filter gains at specified center frequencies. The prediction is implemented with a feedforward neural network having a hidden layer with 20 neurons in the case of the ten-octave graphic equalizer. The band filter coefficients can then be quickly and easily computed using closed-form formulas. This work turns, for the first time, the accurate graphic equalization design into a feedforward calculation without matrix inversion or iterations. The filter gain control using the neural network reduces the computing time by 99.6% in comparison to the least-squares design method it is imitating and contributes an approximation error of less than 0.1 dB. The resulting neurally controlled graphic equalizer will be highly useful in various audio and music processing applications, which require time-varying equalization.**

***Index Terms*—Audio systems, equalizers, feedforward neural networks, IIR filters, supervised learning.**

I. Introduction

GRAPHIC equalization is a common technique in audio and acoustics [1]. Accurate design of a graphic equalizer is more challenging than most users realize [2], [3]. The name of the method suggests that a filter simply produces a magnitude response identical to that controlled by the user, who adjusts the target gains, also called the command gains, at prescribed center frequencies. However, a sufficiently accurate approximation using a single biquadratic IIR (infinite impulse response) filter per command band was only introduced in 2017 [4], [5].

In the analog era, graphic equalizers were constructed using the parallel structure in which all subfilters receive the same input signal [6]. The subfilters were low-order bandpass or resonator filters, but they could not independently adjust the gain at their own band due to interaction between the filters. First digital graphic equalizers were a straightforward discrete-time realization of this basic idea [7]. While the early equalizers remain as powerful tools for modifying and enhancing music and speech signals interactively, in a strict sense their accuracy of approximation is insufficient for high-quality audio.

A second-order parametric equalizer is often used as the band filter in a cascade graphic equalizer [8]–[11]. The parametric equalizing filter is not a bandpass filter, but rather produces a notch or a peak at its pole frequency and approximately unity gain at other frequencies, and is thus well suited to the design of a cascade graphic equalizer in which the total magnitude response is a product of the band-filter responses. However, the interaction problem still remains, and a simplistic implementation in which the gain of each parametric equalizer controlling its own band is set equal to the target gain at that band, is doomed to be very inaccurate [1], [3]. Rämö *et al.* show design examples in which the interaction between neighboring filters causes approximation errors as large as 10 dB in selected cases [2].

A key idea to increase the accuracy in graphic equalizer design is to allow the filter gains to be different from the target gains [1], [3], [4], [12]. The band filter gains must be optimized so that the overall magnitude response of the cascade system matches the target gains at the center frequencies sufficiently accurately, often to within ± 1 dB [5]. In some cases, other frequency points, such as the mid points between each command frequency, are also included in the error calculation [4], [5]. Many approaches for optimizing the subfilters of a cascade [4], [13]–[15] or a parallel graphic equalizer [2], [16]–[18] have been presented. In general, the optimization of a graphic equalizer requires matrix inversion operations or an iteration to be executed every time one of the target gains is varied. This limits the use of accurate graphic equalizers, as the computing power required to run the optimization and filtering simultaneously can be too large for time-varying applications, such as filtering based on approximating human head-related transfer functions (HRTF), which Abel and Berners tackled in [12], or unmasking of music signals, when audio is listened to in a noisy environment [19], [20].

This paper proposes a novel approach to simplify the design of a graphic equalizer. We train a multilayer perceptron [21] to predict the band filter gains from target gains, which are set by the user. The design method proposed recently by Välimäki and Liski [4] opened the door to this approach, as the coefficients of the second-order band filters can be computed in closed-form once the filter gains have been obtained. A least-squares optimization method with one iteration step proposed in [4] is used for creating the necessary training data, i.e., a large number of target gain to filter gain vector pairs. Finally, after the training, the optimization method can be replaced with the multilayer perceptron working in its prediction mode, which leads to a neurally controlled graphic equalizer (NGEQ). The main contribution of this work is to show that the perceptron can learn to be sufficiently accurate for use in high-quality audio graphic equalizers, when predicting filter gains in decibel units. This approach avoids the frequency response evaluation in the training.

The rest of this paper is organized as follows. Section II explains the graphic equalizer design method used in this work

Manuscript received January 14, 2019; revised June 30, 2019 and August 8, 2019; accepted August 9, 2019. Date of publication August 22, 2019; date of current version October 2, 2019. The associate editor coordinating the review of this manuscript and approving it for publication was Dr. Tuomas Virtanen. *(Corresponding author: Vesa Välimäki.)*

The authors are with the Department of Signal Processing and Acoustics, Acoustics Laboratory, Aalto University, FI-02150 Espoo, Finland (e-mail: vesa.valimaki@aalto.fi; jussi.ramo@aalto.fi).

Digital Object Identifier 10.1109/TASLP.2019.2935809

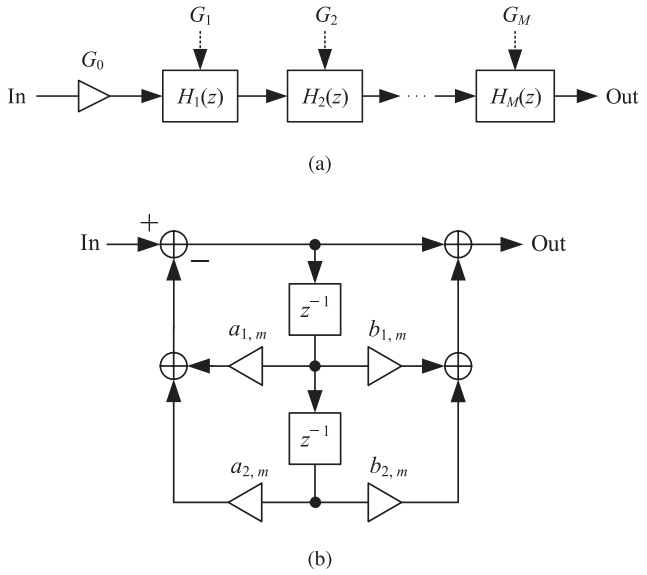

Fig. 1. (a) The cascade graphic equalizer structure with the filter gain controls, G_m, and (b) the biquad IIR filter structure of each band filter $H_m(z)$.

to create the training data for the neural network. Section III discusses how supervised learning can be used to imitate the design. Section IV shows experimental results to validate the proposed design method by comparing them with the original design. Section V concludes the paper.

II. Accurate Graphic Equalizer Design

The filter structure used in the accurate cascade graphic equalizer (ACGE) [4] is shown in Fig. 1(a). Each band filter $H_m(z)$, where $m = 1, 2, \ldots, M$ is the filter index, receives as its input the output of the previous band filter, and finally, the last band filter $H_M(z)$ produces the equalized output signal. The band filters are biquad filters similar to those presented in Fig. 1(b). The next subsection explains how the filter parameters are determined, and Section II-B tackles the optimization of the filter gains G_m.

A. Parametric Equalizer Filter Design

The band filter in the ACGE design by Välimäki and Liski [4] is based on the modified Orfanidis parametric biquadratic equalizing filter [22]. One such filter is used per band, so that $M = 10$ such filters are needed for a standard 10-octave graphic equalizer. Each modified Orfanidis band filter has its own linear gain G_m and center frequency $f_{c,m}$. First, it is necessary to select the center frequencies $f_{c,m}$, which are here set to the recommended octave center frequencies [23]: 31.25 Hz, 62.50 Hz, 125.0 Hz, 250.0 Hz, 500.0 Hz, 1000 Hz, 2000 Hz, 4000 Hz, 8000 Hz, and 16 000 Hz.

A special feature of the modified Orfanidis filter is that, instead of using a standard definition of half-power (-3 dB) bandwidth, it has both a user-defined resonance bandwidth B and a user-defined gain at the bandwidth edges [4], [22]. Välimäki and Liski chose to define the dB gain at the bandwidth edges as

$$g_{B,m} = c\, g_m, \tag{2}$$

where $g_{B,m} = 20\log(G_{B,m})$, $g_m = 20\log(G_m)$, and $0 < c < 1$ is a free parameter defining the proportion of gain at bandwidth edges [4].

The transfer function of this IIR filter can be written as:

$$H_m(z) = b_{0,m}\frac{1 + b_{1,m}z^{-1} + b_{2,m}z^{-2}}{1 + a_{1,m}z^{-1} + a_{2,m}z^{-2}}, \tag{3}$$

where the scaling coefficient is defined as

$$b_{0,m} = \frac{1 + G_m\beta_m}{1 + \beta_m}, \tag{4}$$

where β_m is defined as

$$\beta_m = \sqrt{\frac{|G_{B,m}^2 - 1|}{|G_m^2 - G_{B,m}^2|}}\tan\left(\frac{B_m}{2}\right), \quad \text{when } G_m \neq 1, \tag{5}$$

or as

$$\beta_m = \tan\left(\frac{B_m}{2}\right), \quad \text{when } G_m = 1, \tag{6}$$

the numerator coefficients are

$$b_{1,m} = -2\frac{\cos(\omega_{c,m})}{1 + G_m\beta_m}, \quad b_{2,m} = \frac{1 - G_m\beta_m}{1 + G_m\beta_m}, \tag{7}$$

$$\mathbf{B_0} = \begin{pmatrix}
1.0 & 0.30 & 0.081 & 0.021 & 0.0052 & 0.0013 & 3.2\times10^{-4} & 7.8\times10^{-5} & 1.7\times10^{-5} & 1.9\times10^{-6} \\
0.30 & 1.0 & 0.30 & 0.081 & 0.021 & 0.0052 & 0.0013 & 3.1\times10^{-4} & 6.8\times10^{-5} & 7.5\times10^{-6} \\
0.081 & 0.30 & 1.0 & 0.30 & 0.081 & 0.021 & 0.0052 & 0.0013 & 2.7\times10^{-4} & 3.0\times10^{-5} \\
0.021 & 0.081 & 0.30 & 1.0 & 0.30 & 0.081 & 0.021 & 0.0050 & 0.0011 & 1.2\times10^{-4} \\
0.0053 & 0.021 & 0.081 & 0.30 & 1.0 & 0.30 & 0.080 & 0.020 & 0.0043 & 4.8\times10^{-4} \\
0.0013 & 0.0053 & 0.021 & 0.081 & 0.30 & 1.0 & 0.30 & 0.078 & 0.017 & 0.0019 \\
3.4\times10^{-4} & 0.0014 & 0.0054 & 0.022 & 0.083 & 0.31 & 1.0 & 0.30 & 0.071 & 0.0081 \\
8.2\times10^{-5} & 3.3\times10^{-4} & 0.0013 & 0.0052 & 0.021 & 0.08 & 0.30 & 1.0 & 0.27 & 0.033 \\
2.0\times10^{-5} & 7.9\times10^{-5} & 3.2\times10^{-4} & 0.0013 & 0.0050 & 0.020 & 0.078 & 0.29 & 1.0 & 0.16 \\
4.6\times10^{-6} & 1.9\times10^{-5} & 7.4\times10^{-5} & 3.0\times10^{-4} & 0.0012 & 0.0047 & 0.019 & 0.072 & 0.26 & 1.0
\end{pmatrix} \tag{1}$$

and the denominator coefficients are

$$a_{1,m} = -2\frac{\cos(\omega_{c,m})}{1+\beta_m}, \quad a_{2,m} = \frac{1-\beta_m}{1+\beta_m}, \tag{8}$$

where $\omega_{c,m} = 2\pi f_{c,m}/f_s$ is the normalized center frequency in radians. The sampling rate is $f_s = 44100$ Hz.

The gain factor G_0 in front of the graphic equalizer in Fig. 1(a) is the product of the scaling coefficients of the band filters:

$$G_0 = \prod_{m=1}^{M} b_{0,m}. \tag{9}$$

This way the multiplier related to the scaling factor $b_{0,m}$ can be removed from each band filter section, as can be seen in Fig. 1(b), which saves $M-1$ multiplications in total [24].

The filter bandwidth B_m is defined for each filter as the frequency difference between the neighboring bands, because this allows accurate control of the interaction between the nearest bands. In practice, then, the bandwidth of each band filter is defined as

$$B_m = 1.5\omega_{c,m}. \tag{10}$$

Furthermore, the free design parameter in (2) has been set to $c = 0.30$. These choices have been found to lead to an accurate design [4]. In practice, this means that 30% of the dB-gain of each parametric equalizer leaks to its neighboring center frequencies, as they are defined as the band edges in (10).

However, at high frequencies the magnitude responses of the band filters are asymmetric. This is caused by frequency warping close to the Nyquist limit, and has been reported in previous papers on graphic equalizer design [4], [5]. For this reason, the bandwidth of the final three filters has been set manually to match the center frequency of their lower neighboring filter only. This has led to the following corrections: $B_8 = 1.395\omega_{c,8}$, $B_9 = 1.170\omega_{c,9}$, and $B_{10} = 0.760\omega_{c,10}$.

B. Filter Gain Optimization

The filter gains of the cascade graphic equalizer are optimized with the help of an interaction matrix [4]. The idea is to evaluate the magnitude leakage of each equalizer filter at desired frequency points throughout the frequency range, e.g., at the center frequencies of the graphic equalizer. An initial interaction matrix $\mathbf{B_0}$ can be determined using a predefined prototype gain vector, which has been selected to have all commands set at a fairly large value of 17 dB [4]. In the case of the octave-equalizer design, each row of the interaction matrix will then be set equal to the normalized magnitude response of the m^{th} band filter at the 10 prescribed center frequencies. The normalization is achieved by converting the magnitude response values to decibels and dividing by the prototype gain.

Equation (1) shown at the bottom of the previous page, shows the initial interaction matrix, analyzed at the center frequencies $f_{c,m}$ of the octave equalizer described in Section II-A, with all target gains set to 17 dB. As can be seen, the diagonal values are all 1.0 because of the normalization, which divides each magnitude-response value at the peak by the filter gain, which is the same value. In Section II-A, the parametric equalizing filters were designed so that the gains at neighbouring center frequencies are 0.3 times the peak gain in dB, which can also be seen in (1): the two neighboring values on each side of the diagonal values are 0.30, except on the last three rows, which are different because of asymmetry. Moreover, it is evident that the leakage effect gets smaller and smaller, when the distance from the center frequency of the filter increases to the left or right in (1), as can be expected.

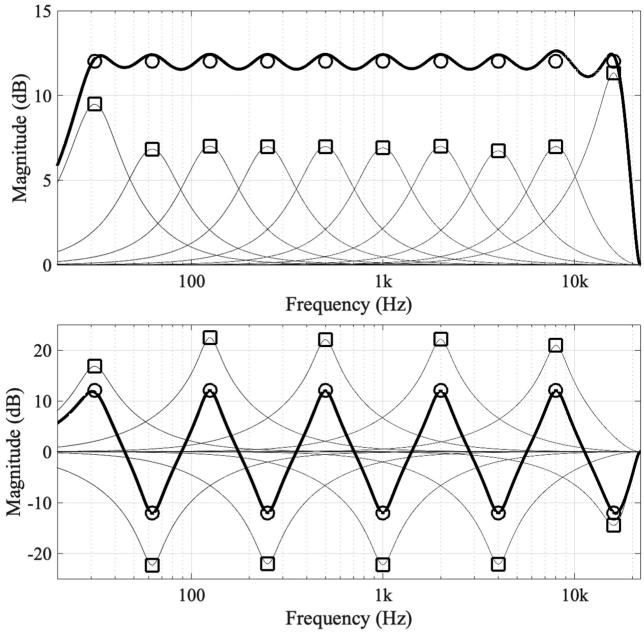

Fig. 2. Two extreme examples of target gains vs. filter gains, where the round markers (○) show the user-set target gains and the square markers (□) are the corresponding optimized filter gains: (upper) all-up at 12 dB and (lower) an extreme zigzag setting. The thin black lines correspond to the individual equalizing filters designed with the optimized filter gains. The thick black line is the magnitude response of the graphic equalizer with these filter gains.

The final filter optimization method in ACGE uses an interaction matrix $\mathbf{B}$ of size $(2M-1) \times M$, where M is the number of band filters and $2M-1$ is the number of design frequencies at which the magnitude response is evaluated, which includes the center frequencies plus their midpoints [4]. The intermediate frequency points are obtained as a geometric mean of each pair of octave frequencies: 44.19 Hz, 88.39 Hz, 176.8 Hz, 353.6 Hz, 707.1 Hz, 1414 Hz, 2828 Hz, 5657 Hz, and 11 314 Hz. In the end, matrix $\mathbf{B}$ has nine additional columns added in between the center-frequency columns shown in (1). This makes the design more accurate, especially between the center frequencies [4].

Furthermore, there is one round of iteration, which uses the initial optimized filter gains to calculate another interaction matrix that is then used for further optimization [4]. This iteration round helps to restrict the approximation error in the magnitude response to be less than ± 1 dB, which was set to be the goal during the design of ACGE [4].

Fig. 2 shows two example cases of the optimized filter gains, where the round markers (○) depict the user-set target gains, i.e., the sliders of a graphic equalizer, and the square markers (□) show the ACGE optimized filter gains. Furthermore, the thin lines show the magnitude responses of the individual equalizer filters, whose peak gains are the optimized filter gain values, while the thick black line shows the magnitude response of the

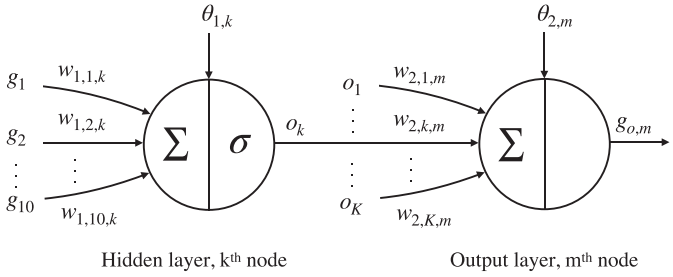

Fig. 3. Structure of individual neurons: (left) a hidden layer unit, which contains a nonlinear sigmoid activation function, and (right) a linear output layer unit, which produces the predicted filter gains.

whole ACGE, which ideally should go through the user-set gain values (○).

The two cases in Fig. 2 illustrate two different target gain configurations, which are known to be hard for graphic equalizers [5]. As can be seen, the optimized values (□) clearly differ from the user-set target gains (○), which highlights the importance of the filter gain optimization. In the top figure all target gains are set to 12 dB. This would create a large overshoot in the total magnitude response without the filter gain optimization that sets all filter gains to be significantly lower than 12 dB, so that their joint effect is correct. An example of such gain buildup, which is at most about 5 dB without filter gain optimization, can be seen in a recent review article (see Fig. 8 in [1]).

The bottom subfigure in Fig. 2 shows a zigzag gain setting in which the target gains are alternating between ± 12 dB, which in turn would not reach the ± 12 dB command points without the filter optimization. It can be seen that the optimized filter gains are much larger than the user-set target gains, while the total magnitude response of the equalizer (thick line) is accurate.

III. Training a Feedforward Neural Network to Control the Filter Gains

This work trains a neural network to calculate the filter gains of a graphic equalizer within the cascaded filter structure described in Section II-A, without the need of the heavy, iterative optimization algorithm described in Section II-B. This is achieved by creating a large set of training data using the ACGE design (Section II) to calculate input-output gain pairs in decibels, see round (○) and square (□) markers in Fig. 2. Training a neural network to predict a small number of filter gain parameters with high accuracy is much easier than building a learning system to optimize five times as many IIR filter coefficients to minimize the magnitude-response error using the backpropagation algorithm! In fact, when only the filter gains are learned, the concept of frequency response (and, thus, the Fourier transform) does not have to be included in the neural network or its training algorithm at all.

A. Network Structure

The structure of the network is 10–K–10, i.e., an input layer consisting of 10 input gains, one hidden layer consisting of K neurons, and an output layer with 10 neurons giving the estimates for the 10 optimized filter gains. Fig. 3 illustrates the individual neurons of the hidden layer and output layer in detail. The leftmost neuron depicts the k^{th} neuron of the hidden layer, where $g_1, g_2, \ldots, g_{10}$ are the input gains, $w_{1,1,k}, w_{1,2,k}, \ldots, w_{1,10,k}$ are the weights of layer 1, $\theta_{1,k}$ is the bias value for the k^{th} neuron in layer 1, σ is the sigmoid activation function, and o_k is the output of the neuron. The output of a hidden layer neuron is calculated as

$$o_k = \sigma\left(\sum_{m=1}^{10} w_{1,m,k} g_m + \theta_{1,k}\right), \tag{11}$$

where the sigmoid activation function σ is implemented with Matlab's `tansig` function, which is equivalent to the hyperbolic tangent function $\tanh(x) = 2/(1+e^{-2x}) - 1$.

The neuron on the right-hand side in Fig. 3 details the m^{th} neuron of the output layer. It takes the outputs $o_1, o_2, \ldots, o_K$ from each of the hidden layer nodes as its inputs, weights them with $w_{2,1,m}, w_{2,2,m}, \ldots, w_{2,K,m}$, and adds them to the bias term $\theta_{2,m}$. The output $g_{o,m}$ of the neural network is calculated as

$$g_{o,m} = \sum_{k=1}^{K} w_{2,k,m} o_k + \theta_{2,m}, \tag{12}$$

which is the optimized filter gain corresponding to the m^{th} band filter.

TABLE I
Input Gains for the Seven Special/Extreme Cases in the Training Data. The +12-dB Gains Have Been Highlighted for Clarity

	Input Gains (dB)									
Case	g_1	g_2	g_3	g_4	g_5	g_6	g_7	g_8	g_9	g_{10}
1	**12**	−12	**12**	−12	**12**	−12	**12**	−12	**12**	−12
2	−12	**12**	−12	**12**	−12	**12**	−12	**12**	−12	**12**
3	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**	**12**
4	−12	−12	−12	−12	−12	−12	−12	−12	−12	−12
5	**12**	−12	−12	**12**	−12	−12	−12	**12**	−12	−12
6	−12	**12**	**12**	−12	**12**	**12**	**12**	−12	**12**	**12**
7	0	0	0	0	0	0	0	0	0	0

B. Training Data

The training data consists of input-output gain pairs, generated using the accurate graphic equalizer design. The input-output gain pairs were generated randomly. In addition to the random data, seven special cases, which are known to be difficult for graphic equalization (listed in Table I), were included in the training data.

The input gains of extreme cases 1 and 3 from Table I are visualized in Fig. 2, where the round markers represent the input gains and the square markers represent the optimized output gains. These examples show that the ACGE method can achieve a good compromise between flatness and selectivity, as it can approximate both a constant high gain setting as well as an extremely fluctuating one, as shown in the upper and lower plots of Fig. 2, respectively. It is also remarkable that in the constant gain case, the filter gains are generally smaller than the target gains while in the zigzag case, the filter gains are generally larger than the targets.

In Matlab, the inputs and outputs of the neural network are scaled in the training phase to have values between -1

and 1. Matlab does this automatically by using a function called `mapminmax`, which scales the data as

$$g'_m = (y_{\max} - y_{\min}) \frac{g_m - x_{\min,m}}{x_{\max,m} - x_{\min,m}} + y_{\min}, \tag{13}$$

$$g_{o,m} = (t_{\max,m} - t_{\min,m}) \frac{g'_{o,m} - y_{\min}}{y_{\max} - y_{\min}} + t_{\min,m}, \tag{14}$$

where $y_{\max} = 1$, $y_{\min} = -1$, $x_{\min,m}$, and $x_{\max,m}$ are the minimum and maximum value of the training data set for the m^{th} input gain, $t_{\min,m}$ and $t_{\max,m}$ are the minimum and maximum value of the training data set for the m^{th} output gain, and g'_m and $g'_{o,m}$ are the scaled input and output gains.

C. Network Training

Before training the neural network, it was decided that the predition error of the network, when compared to ACGE at the command point frequencies, should be less than ± 0.1 dB. This is due to the fact that the reported maximum error of ACGE is approximately ± 0.9 dB [4], [5], and the goal was to create a neural graphic equalizer that has an accuracy within ± 1.0 dB to the user-set target gains.

The network was trained using Matlab's `fitnet` function. The selected training algorithm was the Bayesian regularization backpropagation [25] (`trainbr`), which uses the Levenberg-Marquardt (LM) optimization [26, Ch. 12] to update the weight and bias values. The Bayesian regularization ensures that the resulting network has good generalization capabilities by limiting the sum of squares of the network weights [25], while the LM algorithm provides a desirable compromise between speed and guaranteed convergence of steepest descent [26].

The size of the hidden layer K, as well as the amount of training data needed to successfully train the network was determined by varying the amounts of data and neurons and by training many neural networks with these different parameter settings. The training data set was divided into a training set (70% of the data) and a test set (30%). The training set was used for updating the network weights and biases, while the test set was not used during training. Furthermore, there was no need for a validation data set, which Matlab typically uses with other training algorithms to monitor overfitting of the data during training, since the Bayesian regularization avoids overfitting and improves the model's generalization by constraining the sum of squares of the network weights [25].

Fig. 4 shows the network training performance results with different sizes of training data set for two example network sizes having $K = 19$ (top) and $K = 20$ (bottom) nodes in the hidden layer. The X axis shows the size of the training data set while the Y axis shows the mean squared error (MSE) for the training data and the test data. Both cases, $K = 19$ and $K = 20$, show good results with small error, when training data set size is 600 or larger.

When using Bayesian regularization, it is desired that the training is continued until the neural network converges. To avoid early stopping of the training, which is a known technique to avoid overfitting when not using regularisation, the stopping conditions for the training were set so that the training would stop when the maximum Levenberg-Marquardt μ parameter would reach its default maximum value of 10^{10}, as this is considered to be a good indication that the training algorithm has converged [27]. The adaptive parameter μ is what makes the LM training algorithm so useful. When μ is increased it approaches the steepest descent algorithm with a small step size, and when $\mu = 0$ the algorithm becomes the Gauss-Newton method [26]. The Gauss-Newton method is more accurate and faster near an error minimum, so the goal is to move towards the Gauss-Newton as fast as possible. Thus, the LM algorithm starts with a small value of $\mu = 0.005$. Then, μ is decreased after each successful step by dividing it by $\nu = 10$, so that the training algorithm would move closer to the Gauss-Newton. Furthermore, when the performance function is increased after an unsuccessful step, μ is multiplied by ν. This way the LM training method provides a desirable compromise between the speed of the Gauss-Newton method and the guaranteed convergence of steepest descent. Moreover, after each iteration, the performance function is always decreased.

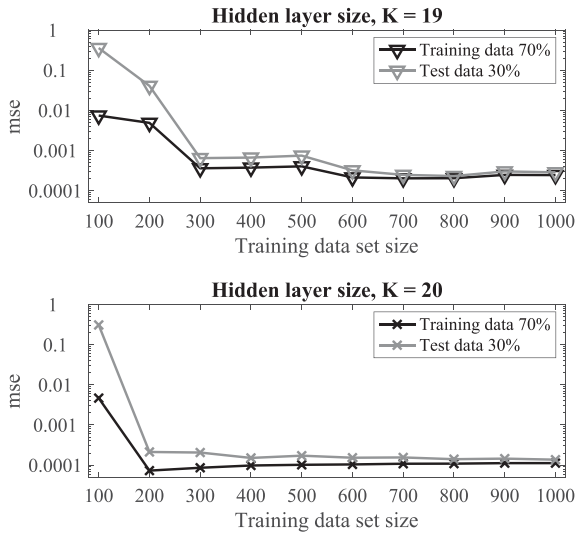

Fig. 4. Remaining MSE as a function of training data set size as a result of training neural networks of two example sizes having (upper) 19 and (lower) 20 hidden layer units. The MSE is generally smaller using the training data set, so the actual validation must be based on the test data.

The maximum number of epochs was set to 10 000 and the MSE performance goal was set to 10^{-5}, which were sufficiently large and small enough, respectively, to allow the training algorithm to converge properly. During the training of different prototype NGEQ neural networks, the training never stopped due to reaching the maximum number of epochs, nor did the networks achieve to have as small MSE as 10^{-5}.

The maximum of the absolute magnitude-response errors between the ACGE and NGEQ methods at the octave center frequencies was calculated. This was found to be a sufficient way for testing, since both equalizers are implemented in the same way, which means that if the error is small at the target gain positions, it is small throughout the whole frequency range. This time, 10 000 random input–output gain pairs were used as validation data. Fig. 5 shows the maximum error as a function of training data set size. The square markers show the errors when $K = 15$, triangular markers when $K = 19$, the crosses

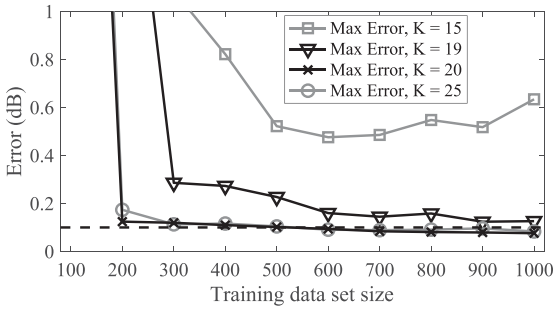

Fig. 5. Maximum error in the EQ magnitude response as a function of the amount of training data, calculated after the training by using a separate, large validation data set of 10 000 random gain settings. The target of 0.1 dB cannot be reached using $K = 19$ hidden units (or less), but with $K = 20$ it is achieved when at least 600 data sets are used.

when $K = 20$, and, finally, the round markers show the errors when $K = 25$. The target error of ± 0.1 dB, set in the beginning of Section III-C, is not met when $K = 19$, as demonstrated in Fig. 5, since the maximum error remains larger at all tested data sizes. However, when $K = 20$, the error falls below 0.1 dB, when the training data set size is larger than 500 input-output pairs. Many other hidden layer sizes K were also tested, but the target error was not reached with a network having less than $K = 20$ hidden layer units, at least, not with moderate training data set sizes. For example, when $K = 15$ (square markers), the error remains above 0.5 dB. Furthermore, when $K = 25$, see the gray line with round markers in Fig. 5, the error does not get any smaller than that of the $K = 20$ case.

D. Resulting Network

Based on the training of the neural network, described in the previous section, the smallest network structure fulfilling the requirements was chosen: It has 10 input nodes, 20 neurons in a single hidden layer, and 10 output nodes. The selected neural network structure is illustrated in Fig. 6, including all the connections of the system. All in all, the neural net has 430 parameters, 200 weights per layer plus 20 bias terms in the hidden layer and 10 in the output layer.

To summarize, the inputs to the net are the user-set target gains in decibels $g_1, g_2, \ldots, g_{10}$, and the outputs are the optimized filter gains in decibels $g_{\mathrm{o},1}, g_{\mathrm{o},2}, \ldots, g_{\mathrm{o},10}$. In between the input and output layers, there is one hidden layer with 20 neurons, which implements a nonlinear multi-input multi-output mapping, as illustrated in Figs. 3 and 6.

E. Prediction Mode: Filter Gain Control

The NGEQ filter gain control can be calculated in matrix form for a set of input gains $\mathbf{g}$ as follows:

$$\mathbf{g}' = 2\frac{\mathbf{g} - \mathbf{x}_{\min}}{\mathbf{x}_{\max} - \mathbf{x}_{\min}} - 1, \tag{15}$$

$$\mathbf{o} = \tanh(\mathbf{W}_1\mathbf{g}' + \boldsymbol{\theta}_1), \tag{16}$$

$$\mathbf{g}'_{\mathbf{o}} = \mathbf{W}_2\mathbf{o} + \boldsymbol{\theta}_2, \tag{17}$$

$$\mathbf{g}_{\mathbf{o}} = (\mathbf{t}_{\max} - \mathbf{t}_{\min})\frac{\mathbf{g}'_{\mathbf{o}} + 1}{2} + \mathbf{t}_{\min}, \tag{18}$$

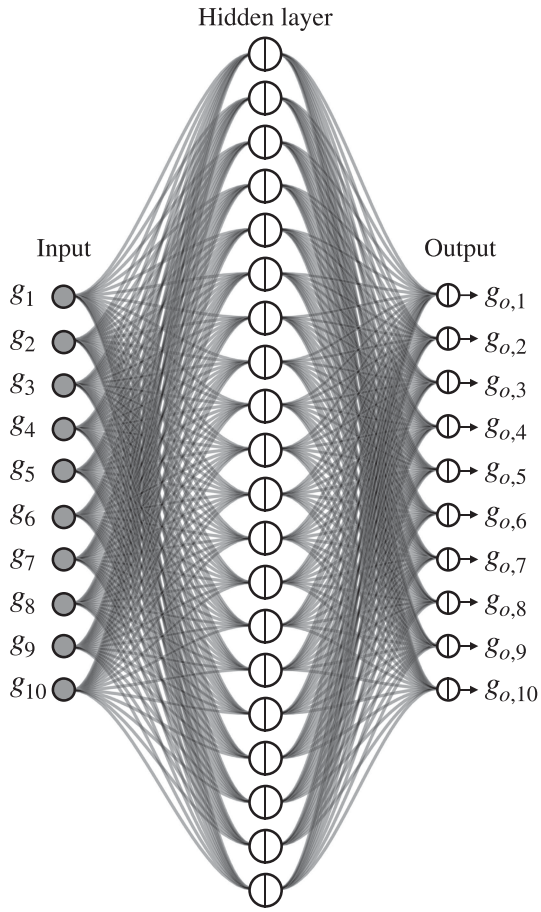

Fig. 6. Neural network structure used in this work, with one hidden layer. The 10 input nodes correspond to the user-defined target gains, and the 10 output nodes correspond the optimized filter gains, both in decibels.

where $\mathbf{W}_1$ is a 20×10 matrix containing the weights $w_{1,m,k}$, $\mathbf{W}_2$ is a 10×20 matrix containing the weights $w_{2,k,m}$, and

$$\mathbf{x}_{\min} = (x_{\min,1}, x_{\min,2}, x_{\min,3}, \ldots, x_{\min,10})^\top, \quad \text{in dB},$$

$$\mathbf{x}_{\max} = (x_{\max,1}, x_{\max,2}, x_{\max,3}, \ldots, x_{\max,10})^\top, \quad \text{in dB},$$

$$\mathbf{g} = (g_1, g_2, g_3 \ldots, g_{10})^\top, \quad \text{in dB},$$

$$\mathbf{g}' = (g'_1, g'_2, g'_3 \ldots, g'_{10})^\top, \quad \in [-1, 1],$$

$$\boldsymbol{\theta}_1 = (\theta_{1,1}, \theta_{1,2}, \theta_{1,3}, \ldots, \theta_{1,20})^\top,$$

$$\mathbf{o} = (o_1, o_2, o_3, \ldots, o_{20})^\top,$$

$$\boldsymbol{\theta}_2 = (\theta_{2,1}, \theta_{2,2}, \theta_{2,3}, \ldots, \theta_{2,10})^\top,$$

$$\mathbf{g}'_{\mathbf{o}} = (g'_{\mathrm{o},1}, g'_{\mathrm{o},2}, g'_{\mathrm{o},3}, \ldots, g'_{\mathrm{o},10})^\top, \quad \in [-1, 1]$$

$$\mathbf{g}_{\mathbf{o}} = (g_{\mathrm{o},1}, g_{\mathrm{o},2}, g_{\mathrm{o},3}, \ldots, g_{\mathrm{o},10})^\top, \quad \text{in dB},$$

$$\mathbf{t}_{\min} = (t_{\min,1}, t_{\min,2}, t_{\min,3}, \ldots, t_{\min,10})^\top, \quad \text{in dB},$$

$$\mathbf{t}_{\max} = (t_{\max,1}, t_{\max,2}, t_{\max,3}, \ldots, t_{\max,10})^\top, \quad \text{in dB}.$$

Equations (15) and (18) have been derived from (13) and (14), respectively. The minimum and maximum values of the training

TABLE II
MINIMUM AND MAXIMUM VALUES OF THE TRAINING DATA, ALL IN DB, FOR SCALING THE INPUTS AND OUTPUTS OF THE NEURAL NETWORK IN (15) AND (18)

m	$\mathbf{x}_{\min}$	$\mathbf{x}_{\max}$	$\mathbf{t}_{\min}$	$\mathbf{t}_{\max}$
1	−12	12	−17.1063	17.3684
2	−12	12	−22.3206	22.3206
3	−12	12	−22.4262	22.4262
4	−12	12	−22.4830	22.4830
5	−12	12	−22.0674	22.0674
6	−12	12	−22.2376	22.2376
7	−12	12	−22.1102	22.1102
8	−12	12	−22.0998	22.0998
9	−12	12	−20.2358	20.2358
10	−12	12	−14.7048	14.7048

data stored in arrays $\mathbf{t_{min}}$, $\mathbf{x_{min}}$ and $\mathbf{t_{max}}$, $\mathbf{x_{max}}$, respectively, are given in Table II.

Finally, the optimized dB-gains $g_{\mathrm{o},m}$ are converted to the linear scale:

$$G_m = 10^{g_{\mathrm{o},m}/20}, \quad \text{for } m = 1, 2, \ldots, 10. \tag{19}$$

The resulting gain factors G_m can be used to compute the filter coefficients with (4)–(9).

IV. COMPARISON

In this section, the results of comparing both the accuracy and the computational cost of the proposed NGEQ method with those of the ACGE method are presented. The final validation data set consists of 10 000 randomized sets of input gain values between −12 dB and +12 dB, in order to cover a vast range of possible gain combinations that users may define.

A. Accuracy

Fig. 7 presents validation plots for the two example cases shown in Fig. 2, where the user-set target gains and the ACGE optimized gains are shown with round and square markers, as previously. The optimized filter gains obtained using the neural network are plotted with crossed markers (×), and ideally, they should lie inside the square markers (☒), as is the case in Fig. 7. Both difficult cases were included in the training data, so the output of the NGEQ algorithm can be expected to be accurate.

The solid line in Fig. 7 is the magnitude response of the NGEQ, which is practically the same as that of the ACGE (not shown), since the approximation error is so small. The subfigure titles report the maximum error with that gain configuration. The error was calculated as the dB-error between the NGEQ and ACGE magnitude responses at the center frequencies. The largest error between the two graphic equalizer designs was 0.006 dB at the 250-Hz command point, when all the command sliders were set to 12 dB, and 0.022 dB at 1 kHz, with the ±12-dB zigzag gain configuration.

Fig. 8 shows a plot similar to Fig. 7, but with random target gains (top) and with the gain setting that produced the largest error (bottom), both taken from the final validation set. These examples were not included in the training data. As can be seen,

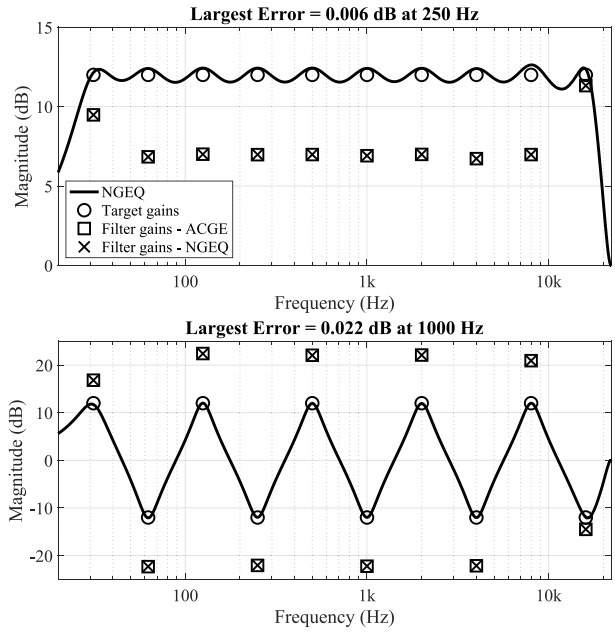

Fig. 7. Validation plots with the examples from Fig. 2, where the crossed markers plot the NGEQ optimized filter gains and the black line shows the magnitude response of the NGEQ. Ideally, the crossed markers should be inside the square markers, illustrating the ACGE optimized filter gains.

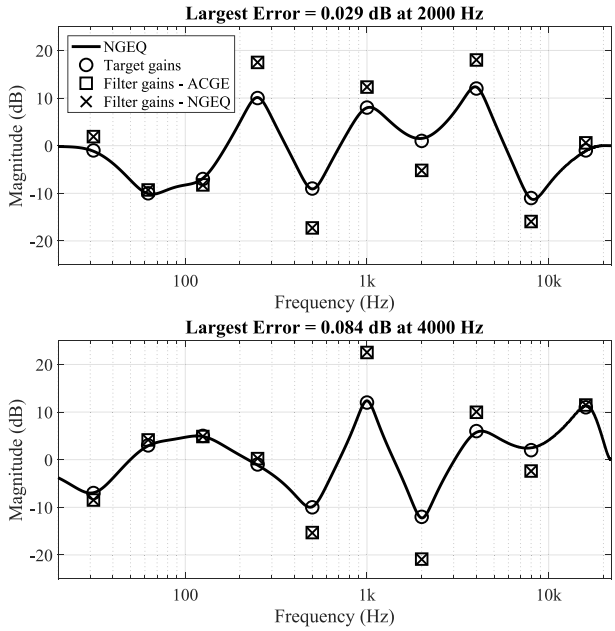

Fig. 8. (Top) Random example from the final validation data set that was not part of the training data, and (bottom) the worst case error of the validation data set, when NGEQ is compared to ACGE.

the NGEQ is highly accurate even with the random gain setting, since all of the NGEQ optimized filter gains (×) overlay on top of the ACGE filter gains (□), with maximum error of 0.029 dB at 2 kHz. Furthermore, even with the worst case error—0.084 dB at 4 kHz—the error is indistinguishable and the final NGEQ response goes through the user-set target gain at 4 kHz.

TABLE III
VALIDATION RESULTS AT COMMAND POINTS FOR THE DATA SET OF 10 000 RANDOM TARGET GAIN SETTINGS

Compared to	Magnitude-Response Error (dB) Min	Max	Average Max
ACGE	−0.083	0.084	0.017
Commands	−0.699	0.693	0.269

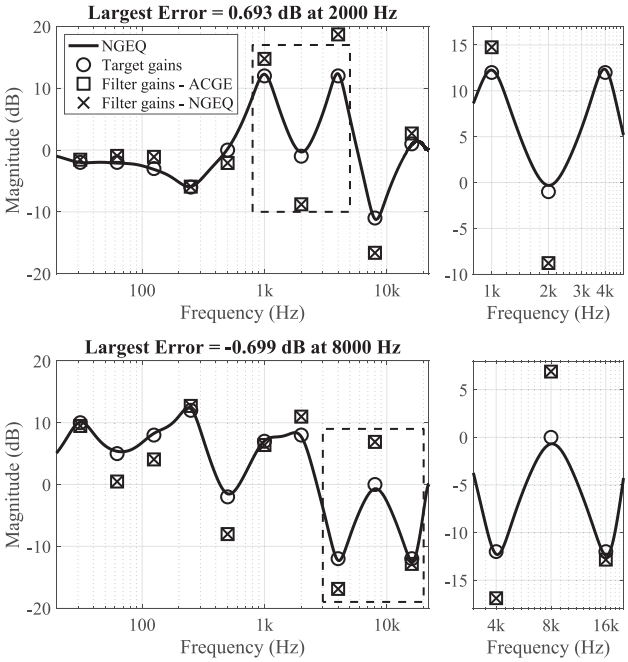

Fig. 9. (Top) Largest overall positive and (bottom) negative error of the final validation set, when the equalizer response was compared to the user-set target gains. The rightmost figures zoom in to the area marked with the dashed-line rectangle to focus on the point of highest error.

Table III shows the minimum and maximum errors for the final validation data set of 10 000 randomized target gain combinations. Furthermore, the last column of Table III shows the average maximum error across all of the 10 000 maximum errors occurring in validation data set. The NGEQ errors calculated against the ACGE at command points are shown in the first row. Furthermore, the errors calculated for the whole frequency range, when compared to ACGE, have similar values as the first row in Table III, meaning the accuracy is good throughout the whole frequency range. The second row in Table III shows the errors when compared against the actual user-set command points. The absolute error from the user-set gain values should be within ± 1 dB, according to the pretraining goal set in Section III-C. This validation test verifies that this is indeed the case, as the magnitude-response error is restricted between ± 0.7 dB, and thus the proposed NGEQ method with $K = 20$ hidden-layer units satisfies the goal of this work.

Fig. 9 reveals where the largest errors occurred in the validation data set, when the magnitude response of the equalizer was compared to the user-set target gains, see Table III bottom row.

TABLE IV
COMPARISON OF COMPUTING TIMES OF THE ACGE AND PROPOSED NGEQ METHODS, AVERAGE OF 10 000 TRIALS. THE FASTEST CASE IN EACH COLUMN IS HIGHLIGHTED

	Gain calculation	Coefficient calculation	Total
ACGE (original)	1700 μs	**24 μs**	1700 μs
NGEQ (proposed)	**6.8 μs**	**24 μs**	**31 μs**

The subfigures on the left-hand side in Fig. 9 show the overall magnitude response, while the subfigures on the right zoom into the position of the largest error (the dashed-lined rectangles).

As can be seen in Fig. 9, both largest errors happen to be similar in nature, where the gains at the adjacent bands have high absolute values, which are very close to each other, while the gain, where the largest error occurs, has a value near zero. In all cases, the filter gains given by the NGEQ are close to those of the ACGE, meaning that the neural network works well and the error mainly comes from the inaccuracy of the original ACGE method used to train the model. The difference between the NGEQ and ACGE filter gains at 2 kHz and 8 kHz was 0.014 dB (top) and 0.006 dB (bottom), respectively.

B. Computational Cost

The computing time for optimizing the filter gains were calculated as an average of the 10 000 validation cases. The results are summarized in Table IV. The NGEQ has the average gain calculation time of about 6.8 μs whereas the ACGE optimization takes 1700 μs on average. This implies that the proposed filter gain control in the NGEQ can calculate 250 gain configurations in the same amount of time it takes for the ACGE to calculate just one.

After the filter gains have been computed, both methods use the same filter design method to calculate the filter coefficients, as described in Section II-A. The calculation of the filter coefficients for 10 parametric equalizing filters takes approximately 24 μs, as indicated in the second column of numbers in Table IV. It is an insignificant amount of the total computing time of the original ACGE design. However, it is remarkable that in the proposed NGEQ method the computing of the filter gains now takes only less than a third of the time it takes to compute the filter coefficients, which is supposed to be an easier problem. The total time consumed in the coefficient update using the proposed method is now 31 μs. It is only slightly more than a sample interval, which is 22.7 μs at the 44.1-kHz sample rate. A small increase in processing speed, which can be obtained by using a faster software implementation and/or a faster computer, such as a graphics-processing unit, would allow coefficient update in real time. This shows that the supervised learning approach is very well suited to the gain calculation in this filter design problem.

The ACGE optimization described in Section II-B requires the calculation and inversion of the interaction matrix and several matrix multiplications. The building of the interaction matrix involves the use of the discrete-time Fourier transform to

evaluate the magnitude response of the parametric equalizing filters at 19 frequency points, which include the ten octave center frequencies and their midpoints. The matrix inversion requires the computing of the Penrose-Moore pseudoinverse for the resulting 19-by-10 interaction matrix, which involves a matrix inversion and three matrix multiplications [4]. It is clear that the proposed neural network based design is much faster and conceptually simpler than the original ACGE optimization method. The neural network based gain control requires two matrix multiplications, 20 evaluations of a nonlinear function, and scaling operations. The matrices are precomputed during the training phase, so they do not contribute to the workload during run time.

V. Conclusion

This paper proposes to use a neural network to speed up the gain control of a graphic equalizer used commonly in audio processing. This was made possible by the recent accurate graphic equalizer design method, which requires the filter gains to be optimized, whereas the filter parameters can be computed in closed form. This work uses the target gain to filter gain vector pairs obtained with the accurate design method as training data for a multilayer perceptron. Testing with different numbers of hidden layer neurons led to the result that 20 neurons are needed for a sufficiently precise prediction of the filter gains in the octave graphic equalizer. The predicted filter gains are then within ± 0.1 dB from those given by the original optimization algorithm. The overall magnitude-response error remains below 1 dB, as is required in high-fidelity audio.

The proposed calculation of the filter gains with a neural network is roughly 250 times faster than the original design method, which requires several matrix operations. It is also the first method to accurately compute the filter coefficients of a graphic equalizer without iterations or matrix inversions. As a result of this work, the design of a graphic equalizer now becomes an easy task, which can be implemented with a few lines of program code and with the help of two weight matrices and bias vectors as well as input and output scaling factors. These will be published online at http://research.spa.aalto.fi/publications/papers/ieee-taslp-ngeq/.

Future work includes further investigations on alternative biquad filter structures and gain optimization schemes, which could be used to create even more accurate training data for the neural network. In the future, the computationally efficient graphic equalizer design method introduced in this paper can be used in various real-time audio applications, which require time-varying high-order equalization, such as HRTF filtering controlled by the user's orientation (head tracking) in a virtual/augmented reality scenario, or unmasking of music signals when listening takes place in heavy ambient noise.

Acknowledgment

The authors would like to thank B. Bank, J. Liski, J. O. Smith, and A. Wright for helpful discussions.

References

[1] V. Välimäki and J. D. Reiss, "All about audio equalization: Solutions and frontiers," *Appl. Sci.*, vol. 6, no. 129, pp. 1–46, May 2016.
[2] J. Rämö, V. Välimäki, and B. Bank, "High-precision parallel graphic equalizer," *IEEE/ACM Trans. Audio, Speech, Lang. Process.*, vol. 22, no. 12, pp. 1894–1904, Dec. 2014.
[3] R. J. Oliver and J.-M. Jot, "Efficient multi-band digital audio graphic equalizer with accurate frequency response control," in *Proc. Audio Eng. Soc. 139th Conv.*, New York, NY, USA, Oct. 2015, Paper 9406.
[4] V. Välimäki and J. Liski, "Accurate cascade graphic equalizer," *IEEE Signal Process. Lett.*, vol. 24, no. 2, pp. 176–180, Feb. 2017.
[5] J. Liski and V. Välimäki, "The quest for the best graphic equalizer," in *Proc. Int. Conf. Digit. Audio Effects*, Edinburgh, U.K., Sep. 2017, pp. 95–102.
[6] R. A. Greiner and M. Schoessow, "Design aspects of graphic equalizers," *J. Audio Eng. Soc.*, vol. 31, no. 6, pp. 394–407, Jun. 1983.
[7] Motorola, Inc., "Digital stereo 10-band graphic equalizer using the DSP56001," Chicago, IL, USA, Appl. Note APR2/D, 1988.
[8] S. A. White, "Design of a digital biquadratic peaking or notch filter for digital audio equalization," *J. Audio Eng. Soc.*, vol. 34, no. 6, pp. 479–483, Jun. 1986.
[9] P. A. Regalia and S. K. Mitra, "Tunable digital frequency response equalization filters," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-35, no. 1, pp. 118–120, Jan. 1987.
[10] T. van Waterschoot and M. Moonen, "A pole-zero placement technique for designing second-order IIR parametric equalizer filters," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 15, no. 8, pp. 2561–2565, Nov. 2007.
[11] J. D. Reiss, "Design of audio parametric equalizer filters directly in the digital domain," *IEEE Trans. Audio, Speech, Lang. Process.*, vol. 19, no. 6, pp. 1843–1848, Aug. 2011.
[12] J. S. Abel and D. P. Berners, "Filter design using second-order peaking and shelving sections," in *Proc. Int. Comput. Music Conf.*, Miami, FL, USA, Nov. 2004, pp. 1–4.
[13] M. Holters and U. Zölzer, "Graphic equalizer design using higher-order recursive filters," in *Proc. Int. Conf. Digit. Audio Effects*, Montreal, QC, Canada, Sep. 2006, pp. 37–40.
[14] J. Rämö and V. Välimäki, "Optimizing a high-order graphic equalizer for audio processing," *IEEE Signal Process. Lett.*, vol. 21, no. 3, pp. 301–305, Mar. 2014.
[15] S. Prince and K. R. S. Kumar, "A novel Nth-order IIR filter-based graphic equalizer optimized through genetic algorithm for computing filter order," *Soft Comput.*, vol. 23, no. 8, pp. 2683–2691, Apr. 2019.
[16] S. Tassart, "Graphical equalization using interpolated filter banks," *J. Audio Eng. Soc.*, vol. 61, no. 5, pp. 263–279, May 2013.
[17] Z. Chen, G. S. Geng, F. L. Yin, and J. Hao, "A pre-distortion based design method for digital audio graphic equalizer," *Digit. Signal Process.*, vol. 25, pp. 296–302, Feb. 2014.
[18] B. Bank, J. A. Belloch, and V. Välimäki, "Efficient design of a parallel graphic equalizer," *J. Audio Eng. Soc.*, vol. 65, no. 10, pp. 817–825, Oct. 2017.
[19] J. Rämö, V. Välimäki, and M. Tikander, "Perceptual headphone equalization for mitigation of ambient noise," in *Proc. IEEE Int. Conf. Acoust., Speech, Signal Process.*, Vancouver, BC, Canada, May 2013, pp. 724–728.
[20] V. Välimäki, A. Franck, J. Rämö, H. Gamper, and L. Savioja, "Assisted listening using a headset: Enhancing audio perception in real, augmented, and virtual environments," *IEEE Signal Process. Mag.*, vol. 32, no. 2, pp. 92–99, Mar. 2015.
[21] C. M. Bishop, *Pattern Recognition and Machine Learning*. New York, NY, USA: Springer, 2006.
[22] S. J. Orfanidis, *Introduction to Signal Processing*. Upper Saddle River, NJ, USA: Prentice-Hall, 1996.
[23] *Acoustics—Preferred Frequencies*, ISO 266:1997, International Organization for Standardization, 1997.
[24] J. Liski, B. Bank, J. O. Smith, and V. Välimäki, "Converting series biquad filters into delayed parallel form: Application to graphic equalizers," *IEEE Trans. Signal Process.*, vol. 67, no. 14, pp. 3785–3795, Jul. 2019.
[25] F. D. Foresee and M. T. Hagan, "Gauss-Newton approximation to Bayesian learning," in *Proc. IEEE Int. Conf. Neural Netw.*, Houston, TX, USA, Jun. 1997, pp. 1930–1935.
[26] M. T. Hagan, H. B. Demuth, M. H. Beale, and O. D. Jesús, *Neural Network Design*, 2nd ed. E-Book, 2014. [Online]. Available: http://hagan.okstate.edu/nnd.html
[27] The MathWorks, Inc., Improve shallow neural network generalization and avoid overfitting, 2019. [Online]. Available: https://se.mathworks.com/help/deeplearning/ug/improve-neural-network-generalization-and-avoid-overfitting.html

Vesa Välimäki (S'90–M'92–SM'99–F'15) received the M.Sc. degree in technology and the Doctor of Science degree in technology, both in electrical engineering, from the Helsinki University of Technology (TKK), Espoo, Finland, in 1992 and 1995, respectively.

He was a Postdoctoral Research Fellow with the University of Westminster, London, U.K., in 1996. In 1997–2001, he was a Senior Assistant (cf., Assistant Professor) with TKK. In 2001–2002, he was a Professor of Signal Processing with the Pori unit of the Tampere University of Technology, Pori, Finland. In 2006–2007, he was the Head of the TKK Laboratory of Acoustics and Audio Signal Processing. In 2008–2009, he was a Visiting Scholar at Stanford University. He is currently a Full Professor of Audio Signal Processing and the Vice Dean for Research in electrical engineering with Aalto University, Espoo, Finland. His research interests include digital filter design, audio effects processing, artificial reverberation, sound synthesis, and signal processing for headphones and loudspeakers.

Prof. Välimäki is a Fellow of the Audio Engineering Society and a Life Member of the Acoustical Society of Finland. In 2007–2013 he was a Member of the Audio and Acoustic Signal Processing Technical Committee of the IEEE Signal Processing Society and is currently an Associate Member. He is a Founding Member of the EURASIP Special Area Team in acoustic, sound, and music signal processing. He served as an Associate Editor for the IEEE SIGNAL PROCESSING LETTERS in 2005–2009 and for the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING in 2007–2011. He was in the Editorial Board of the *Research Letters in Signal Processing* and of the *Journal of Electrical and Computer Engineering*. He was the Lead Guest Editor of a special issue of the IEEE SIGNAL PROCESSING MAGAZINE in 2007 and of a special issue of the IEEE TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING in 2010. He was a Guest Editor of the special issues of the IEEE SIGNAL PROCESSING MAGAZINE on signal processing techniques for assisted listening in 2015 and on music signal processing in 2018–2019. He has been a Guest Editor of two special issues published in *Applied Sciences*, one in 2016 and another in 2017–2018. Since 2015, he has been a Senior Area Editor of the IEEE/ACM TRANSACTIONS ON AUDIO, SPEECH, AND LANGUAGE PROCESSING. In 2008, he was the Chair of the International Conference on Digital Audio Effects (DAFX). In 2017, he was the Chair of the International Conference on Sound and Music Computing.

Jussi Rämö received the M.Sc. degree in communication engineering from the Helsinki University of Technology, Espoo, Finland, in 2009, majoring in acoustics and audio signal processing, and the Doctor of Science degree in technology from Aalto University, Espoo, Finland, in 2014.

His doctoral dissertation dealt with equalization techniques for headphone listening. He was an Industrial Postdoctoral Researcher with Aalborg University, Aalborg, Denmark, in 2015–2017, responsible for a project for Bang & Olufsen concentrating on developing psychoacoustical models for sound-zone environments in domestic rooms. The project was partly funded by the Innovation Fund Denmark. He is currently a Postdoctoral Researcher with Aalto University, Espoo, Finland. His research interests include psychoacoustics, equalizers, headphone signal processing, and perceptually motivated signal processing, especially for mobile applications.

Dr. Rämö was a member of the organizing committee of the Audio Engineering Society 51st International Conference on Loudspeakers and Headphones, Helsinki, Finland, August 2013.