Audio Engineering Society

Convention Paper

Presented at the 124th Convention
2008 May 17–20 Amsterdam, The Netherlands

Graphical Control of a Parametric Equalizer

Jörn Loviscach[1]

[1] *Hochschule Bremen (University of Applied Sciences), 28199 Bremen, Germany*

Correspondence should be addressed to Jörn Loviscach (`joern.loviscach@hs-bremen.de`)

ABSTRACT

Graphic equalizers allow the user to define a filter's magnitude response virtually free of restrictions. Parametric equalizers are much more limited. However, they offer some vital advantages over graphic equalizers, such as consuming less computational power and operating minimally invasively with naturally soft magnitude and phase responses. This work aims at combining the best of both worlds: It presents a range of methods to control a digital parametric equalizer graphically through a curve or a collection of anchor points. While the user is editing the graphical input, an optimization process runs in the background and adjusts the equalizer's parameters to reflect the input. In addition, the number of bands and their type (shelving/peak) can be adjusted automatically to produce a simple solution.

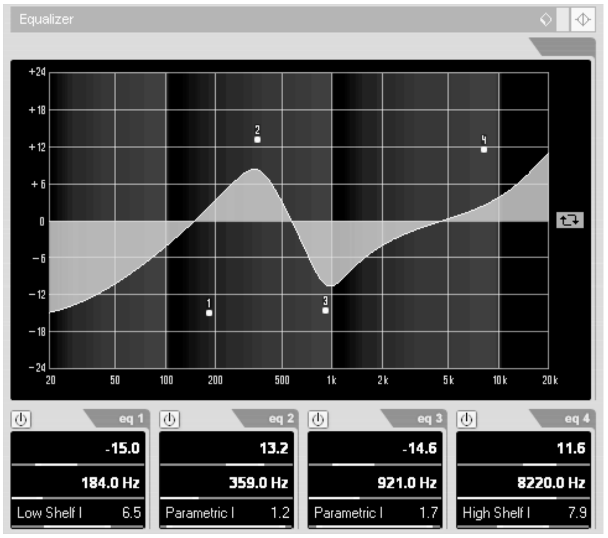
1. INTRODUCTION

Commonly, digital parametric equalizers offer a display showing the magnitude response that results from the current setting. Many products also allow the user to make graphical adjustments of the center/edge frequencies and the boost/cut gains of the different bands through a collection of dots on a 2D display. However, the resulting curve passes only rarely through the specified points, see Figure 1. Entering a given curve with such a user interface requires several iterations, because adjusting one dot will offset the curve elsewhere.

This paper introduces methods that offer a graphical but at the same time precise control. The sound engineer can freely choose from this spectrum of methods in one user interface, see Figure 2. The classical parameters of the equalizer are accessible on demand; the presented system merely assists in setting them.

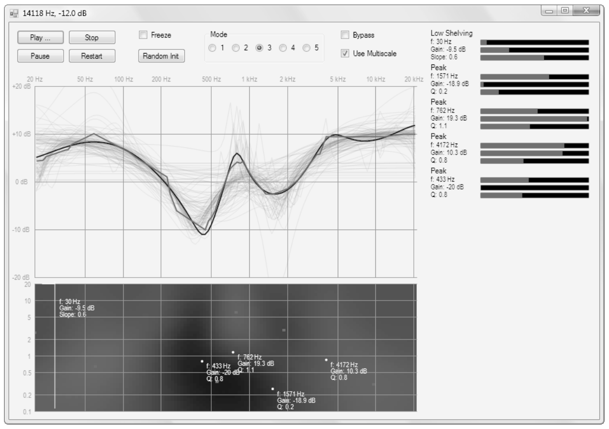
The different methods are realized as modes in the software prototype:

1. The user places anchor points at center/edge frequencies and levels and sets the band type



eq 1	eq 2	eq 3	eq 4
-15.0	13.2	-14.6	11.6
184.0 Hz	359.0 Hz	921.0 Hz	8220.0 Hz
Low Shelf I 6.5	Parametric I 1.2	Parametric I 1.7	High Shelf I 7.9

Fig. 1: Many digital parametric equalizers such as this in Steinberg Cubase offer interactive control through (off-curve) dots in the magnitude response diagram.



Mode: 1 2 [✓ 3] 4 5

[✓ Use Multiscale]

Low Shelving
f: 30 Hz
Gain: -9.5 dB
Slope: 0.6

Peak
f: 1571 Hz
Gain: -18.9 dB
Q: 0.2

Peak
f: 762 Hz
Gain: 19.3 dB
Q: 1.1

Peak
f: 4172 Hz
Gain: 10.3 dB
Q: 0.8

Peak
f: 433 Hz
Gain: -20 dB
Q: 0.8

Fig. 2: Depending on the system's mode, the user can place anchor points or draw freehand lines. The current population of the genetic algorithm is shown behind the user input (lighter color) and the curve of the currently best setting (darker color). Its frequency and Q values are displayed on top of the multiscale analysis (bottom). Standard controls are offered on the right hand side.

(peak or low or high shelving) and the Q or slope values.

2. The user places a small number of anchor points along the intended magnitude response curve.

3. The user draws a freehand curve to determine the intended magnitude response.

4. The user draws two freehand curves to determine a lower and an upper bound for the magnitude response.

5. The user draws a set of freehand strokes and defines their relative importance.

In the first mode, the optimization adjusts only the cut/boost gains; for the latter four modes, the optimization adjusts all parameters, including the number and type of bands. The optimization aims at reproducing the user's input as an actual magnitude response of the equalizer. This will not always be possible exactly, however. For instance, the settings for boost/cut gains in mode 1 may be too extreme; or the input curve in mode 3 may be too complex. In such as case, the optimization aims at a compromise. Modes 4 and 5 allow the user to be more specific as to which part of the input possesses which priority.

Related functions are known from computer graphics: Some vector graphics software lets the user directly drag a point anywhere on a Bézier curve; and 3D software computes "inverse kinematics" enabling the user to animate virtual characters by dragging their limbs with the mouse. However, optimizing the settings of a parametric equalizer faces several difficulties: The resulting curve depends on the settings in a complex fashion; architectural changes (number and types of bands) have to be taken into account; and the user's input will almost always be either under- or overconstraining. Thus, a genetic algorithm offers itself for the optimization. The settings and the architecture of the equalizer can easily be "mutated" and "recombined." An error measure quantifies how well the resulting magnitude response reproduces the user's input. If this is mimicked closely enough, also the number of bands used may enter the error measure. Thus, simpler solutions with fewer bands can be favored if they can achieve the user's wishers.

There is a considerable body of research on the optimization of filters, ranging from analytical methods such as Yule-Walker to search methods such as genetic algorithms. The work presented here is unique in the following two respects:

- It does not address a general filter but a standard parametric equalizer. Thus, it still allows the user to override the result by adjusting easily comprehensible parameters.
- The optimization is not an offline process but proceeds in real time and with immediate audio feedback as the user adds, moves or deletes anchor points or draws or erases curves.

The presented prototype realizes a parametric equalizer modeling the EQ III found in Digidesign Pro Tools: It uses up to five bands, the first and last of which can be switched to shelving mode. EQ III restricts every band to a certain frequency range; this feature is not carried over to the model.

On a standard PC, the optimization process runs at interactive speed. Only for highly complex input curves it may take several seconds to find an appropriate solution. This quick response is enabled by seeding the genetic algorithm with plausible candidates. Curves entered by the user are for instance subjected to a multiscale analysis to find the positions and widths of peaks and valleys, which translate into approximate center/edge frequencies and Q settings. The optimization occurs only during editing, so that the computational cost does not depend on the total number of equalizers in the system, since only one will be affected at a time.

The automatic design of general IIR filters has been researched into for decades. This work, however, focuses on a restricted set of filters: those that occur in a standard parametric equalizer. In a similar vein, Ramos and Lopez [1] construct an IIR filter as a chain of second-order sections (bands), which are peak, high-pass or low-pass filters offering three or two parameters. The optimization aims at minimizing the L_1 distance determined with a logarithmic frequency axis. The strategy consists in finding the most important peak or dip, correcting it by a second-order section, the finding the second-most important peak or dip and so on. This approach is related to the approach pursued in this paper; however, this paper focuses on interactive operation and applies a different optimization method. Genetic methods such as used in this work are known to be highly effective in filter design [2] and are prominent in recent research on this topic [3, 4, 5, 6].

2. OPTIMIZATION STRATEGY

The genetic optimization process is based on a population of genotypes, each representing the complete settings of the parametric equalizer. One special genotype corresponds to using no filter at all; all other genotypes comprise one to five bands, where each band can be a peak filter, a low- or a high-shelving filter. Only one of each of the latter types may occur per equalizer setting. In addition to the filter parameters, every setting contains an overall gain level. The filters are implemented according to Bristow-Johnson [7], with controls for frequency, gain and Q for a peak filter band or slope for a shelving filter band.

The user inputs anchor points that describe the intended magnitude response. The freehand curves entered in modes 3 through 5, too, are stored and edited as sets of points corresponding to 120 logarithmically spaced frequency bands (approximately semitones). The basic error measure is the sum of squared level differences between the target and the actual magnitude response. Using squares (that is, an L_2 optimization opposed to, say, an L_1 or L_∞ optimization) allows computing the optimal overall gain level as the average of the level differences at all anchor points.

A large population size helps to evade a local but non-global optimum; it will, however, also slow down the convergence, as more individuals have to be taken care of. In preliminary tests, the number of 100 genotypes comprising all choices of structures (number and types of bands) turned out to offer a good balance between the convergence rate and the response time for large edits. For detailed results on the speed see Section 4.

The choice of the actual optimization method turned out to be uncritical. The prototype software applies ten rounds of random changes to every genotype and keeps the improving changes. (Whereas this random search may be accelerated through a gradient descent, computing the gradient of the magnitude response would highly increase the method's complexity.) Then, five tournaments are done, each between five random genotypes. The worst genotype in a tournament is replaced by settings created using data from the anchor points or the multiscale analysis, see Section 3. In addition, random bands of the second-worst genotype are replaced by bands from the best one (gene crossover).

The best current setting is also used for the real-time audio processing that occurs in parallel to the optimization. The user can also freeze the optimization and change the setting in the traditional way: through sliders.

3. CONTROL MODES

3.1. Mode 1: Automatic Gain Adaption

The first mode for pointwise control closely resembles the standard interface for a parametric equalizer: The user sets the number of filter bands, their type (peak filter or low or high shelving filter) and their Q or slope. He or she places anchor points in a 2D diagram to determine the bands' frequencies and boost/cut gains. In contrast to the standard interface, however, the anchor points do *not* directly specify the boost/cut gain. Rather, the automatic optimization tries to pick such gain values that the magnitude response curve passes through all anchor points, see Figure 3. Since there is one unknown boost/cut gain per band plus the unknown overall gain, the user has to place one additional anchor point. It does not represent a band of the equalizer but indicates a reference level to be aimed at.

In this mode, the optimization process minimizes the sum of the squared level differences between the actual and the intended magnitude response at the anchor points. The user may place the anchor points in such a way that there is no exact solution, see Figure 4. Then the optimization will still find an optimal but not exact solution. In principle, the genetic optimization method is too sophisticated for this mode. However, employing the same algorithm for all five modes facilitates integration and testing.

3.2. Mode 2: Connecting the Dots

The second type of pointwise control allows the user to place an arbitrary number of anchor points in the frequency/level diagram. The optimization process

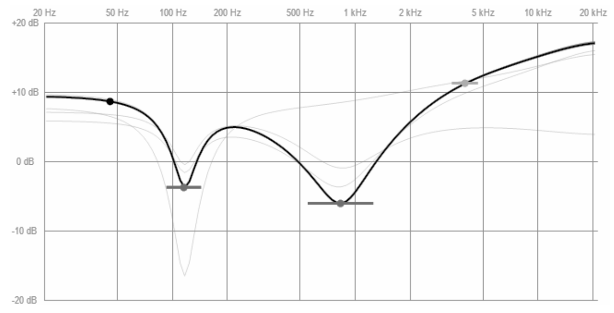


Fig. 3: In mode 1, the user can place one reference dot and one additional dot per band. The band types are indicated by color. Frequencies and gains can be edited by dragging the dot with the mouse. Q and slope values are indicated by horizontal bars and can be changed through the scroll wheel; a click on the scroll wheel changes a band's type.

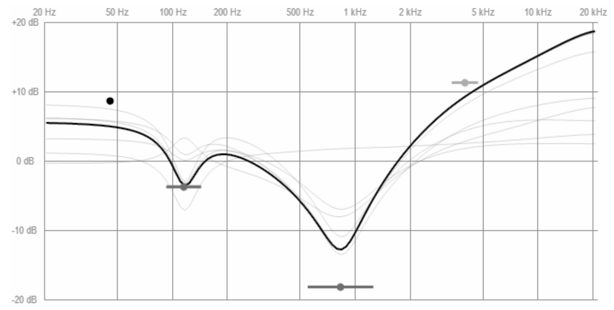


Fig. 4: If the user places the dots in mode 1 in such a way that there is no exact solution, the optimization aims at finding the best existing setting.

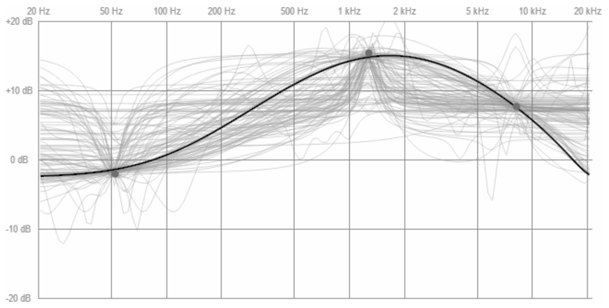


Fig. 5: In mode 2, the system attempts to find the simplest setting the curve of which passes through all prescribed points, such as the single peak filter constructed from three anchors shown here.

tries to determine values for all parameters of the equalizer, including the number and types of filter bands, see Figure 5. When the optimization process needs to create a new random filter, it uses one of the frequencies of the anchor points and sets a random gain and a random Q or slope.

The error measure is defined in such a way that simple solutions are preferred whenever they meet the user's input with sufficient accuracy. Let s be the sum of the squared level differences of the actual and the intended magnitude response at the anchor points. If $\sqrt{s}$ is larger than 1.0 dB, the error measure is $s + 1$. If, however, the user's input is met so well that $\sqrt{s} \leq 1.0$ dB, the error is set to $1 - (1 + c)^{-1}$, where $c \in [0, \infty)$ is a measure of the equalizer's complexity. c itself is a sum of complexities that are defined per filter band. A peak filter is rated with a higher complexity than a shelving filter. In addition, a filter band's complexity grows as its cut/boost gain or Q or slope settings get large.

3.3. Mode 3: Freehand Curve

This mode comes closest to a standard graphical equalizer: The user draws a curve in the frequency/level diagram that specifies the intended magnitude response, see Figure 6. The error measure is computed as follows: Let s be the average squared level difference between the actual and the intended magnitude response. If the magnitude of the difference is less than 1.0 dB at all frequencies, the error is set to $s/n + 1$, where n is the number of bands, otherwise it is set to $s/n + 1 - (1 + c)^{-1}$, paralleling the use of the complexity measure in mode 2. There is a small difference, however: In mode 3, s is incorporated in the error measure regardless of whether the found solution is good or not. This enforces further optimization of a solution that deviates from the intended curve only by a tiny amount but over a large range of frequency bands.

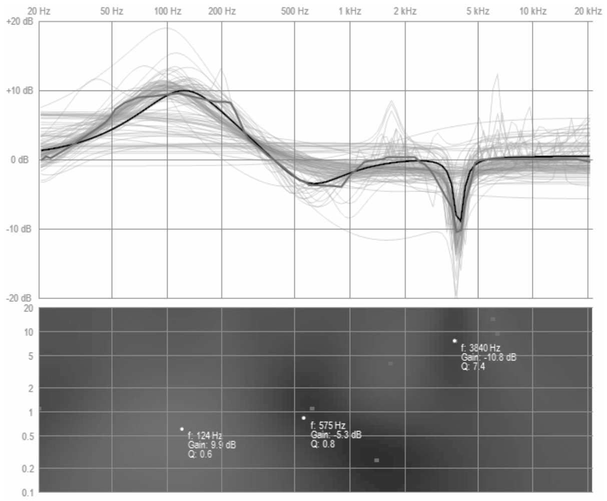


Fig. 6: In mode 3, the user edits a freehand curve. The local extrema of a multiscale analysis of that curve (bottom, darker dots) are used to guess initial settings. The best current setting is shown in this diagram as well (white dots).

Whenever the optimization process requests a new genotype, no random collection of parameters is employed. Instead, to speed up the search process, a secondary processing thread generates plausible sets of parameter values from the curve that the user has sketched. To this end, this curve h, which maps logarithmically spaced frequencies k to levels $h(k)$ measured in dB, is subjected to a multi-resolution analysis in this doubly-logarithmic domain, see the lower part of Figure 6.

This analysis is based on the Difference of Gaussians wavelet:

$$\phi(k) = \frac{1}{\sqrt{2\pi}} \left(2^{1/2} \exp(-k^2) - 2^{-1/2} \exp(-0.25k^2) \right),$$

For a collection of frequencies k and widths σ we form

$$W(k, \sigma) = \sigma^{-1/2} \int h(k') \phi\left(\frac{k' - k}{\sigma} \right) dk'.$$

This integral (actually a sum over the 120 frequency bands) can be interpreted as the—positive or negative—bump intensity of the curve h with a width σ at the frequency band k.

The factor $\sigma^{-1/2}$ in front of the integral ensures that $W(k,\sigma)$ attains a maximum value at $k = k_0$ and $\sigma = \sigma_0$ when $h(k')$ equals a single bump $\phi((k'-k_0)/\sigma_0)$. To see this, note that in this case

$$\begin{aligned}
&\left.\frac{\partial W(k_0,\sigma)}{\partial\sigma}\right|_{\sigma_0} \\
&= \left.\frac{d}{d\sigma}\right|_{\sigma_0}\left(\sigma^{-1/2}\int\phi\left(\frac{k'-k_0}{\sigma_0}\right)\phi\left(\frac{k'-k_0}{\sigma}\right)\right)dk' \\
&= -\frac{1}{2}\sigma_0^{-3/2}\int\phi\left(\frac{k'-k_0}{\sigma_0}\right)\phi\left(\frac{k'-k_0}{\sigma_0}\right)dk' \\
&\quad +\sigma_0^{-1/2}\int\phi\left(\frac{k'-k_0}{\sigma_0}\right)\phi'\left(\frac{k'-k_0}{\sigma_0}\right)\frac{-k'}{\sigma_0^2}\,dk' \\
&= -\frac{1}{2}\sigma_0^{-3/2}\int\left(\phi\left(\frac{k'-k_0}{\sigma_0}\right)\right)^2 dk' \\
&\quad -\sigma_0^{-5/2}\int k'\frac{\sigma_0}{2}\frac{d}{dk'}\left(\phi\left(\frac{k'-k_0}{\sigma_0}\right)\right)^2 dk' \\
&= -\frac{1}{2}\sigma_0^{-3/2}\int\left(\phi\left(\frac{k'-k_0}{\sigma_0}\right)\right)^2 dk' \\
&\quad +\frac{1}{2}\sigma_0^{-3/2}\int\left(\phi\left(\frac{k'-k_0}{\sigma_0}\right)\right)^2 dk' \\
&= 0,
\end{aligned}$$

where partial integration is used in the penultimate step. In a similar fashion, one can verify that

$$\begin{aligned}
\left.\frac{\partial^2 W(k_0,\sigma)}{\partial\sigma^2}\right|_{\sigma_0} &< 0, \\
\left.\frac{\partial W(k,\sigma_0)}{\partial k}\right|_{k_0} &= 0, \\
\left.\frac{\partial^2 W(k,\sigma_0)}{\partial k^2}\right|_{k_0} &< 0, \\
\left.\frac{\partial^2 W(k,\sigma)}{\partial\sigma\,\partial k}\right|_{\sigma_0,k_0} &= 0.
\end{aligned}$$

Every local maximum and minimum of $W(k,\sigma)$ can be related to a peak filter: k defines a frequency; σ represents the width of the peak, which translates to a Q value. The height of the maximum or width of the minimum is used to determine the gain. To compensate for the factor $\sigma^{-1/2}$ in $W(k,\sigma)$ and for the decay of the integral with decreasing width, the gain is set to $\sigma^{-1/2}W(k,\sigma)$ times an appropriate constant.

The local extrema in the multiscale analysis are marked in the display (see the lower half of Figure 6) and serve as a repository to build new settings from, if required by the optimization.

3.4. Mode 4: Upper and Lower Curve

Mimicking a traditional way of specifying the requirements on a filter, mode 4 lets the user draw a lower and an upper bound for the magnitude response, see Figure 7. New genotypes are created from the multiscale analysis of the average of the upper and the lower curve.

As before, the total error measure is a sum over partial errors for all frequency bands. If the curve stays within the prescribed corridor at a certain frequency band, the partial error for this band is the squared difference of the actual response and the midpoint between the minimum and maximum curve. If the curve leaves the prescribed corridor, the partial error is set to a large penalty factor times the square of the distance between the actual response and the corridor's closest border. This forces the curve to stay within the prescribed region, but at the same time favors incremental improvements. The error measure favors a solution in which fewer bands are outside of the corridor.

3.5. Mode 5: Freehand Strokes plus Importance

The final mode is based on the idea that the user may not want to specify the response everywhere but only in critical regions. In this mode, the user can draw freehand curves and define their relative importance, see Figure 8. In the prototype this occurs through pressing the number keys 1 through 9; a more natural option could be to use the pressure of a stroke on a graphics tablet. Curves can also be erased. If the user draws a curve at a frequency at which there is already another curve, the latter will be removed. This mode, too, employs the multiscale analysis to initialize new genotypes.

Similar to mode 3, also mode 5 incorporates pressure to reduce the complexity of the filter. However, the error threshold below which this complexity reduction takes place is chosen differently: To incorporate

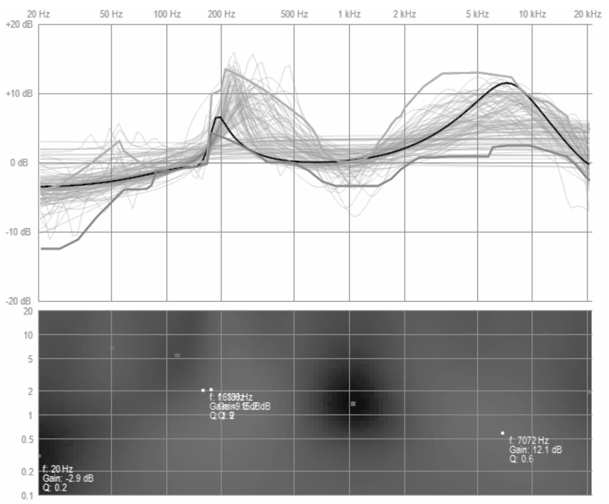


Fig. 7: In mode 4, two curves form the upper and the lower bound. The multiscale analysis is built from their average.

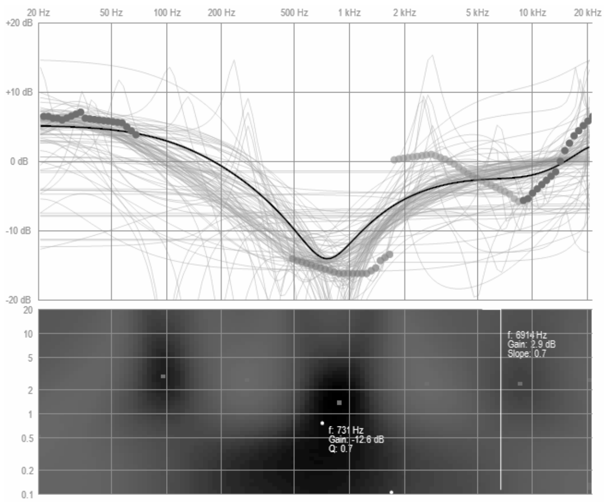


Fig. 8: In mode 5, the user can specify freehand strokes plus their importance. The diagram indicates the importance by the degree of opacity of the dots that form the curve.

the importance values, we compute a sum of squared errors, weighted by the importance assigned by the user. If this weighted sum is less than $(3\,\mathrm{dB})^2$, the filter's complexity is taken into account in the error measure.

4. RESULTS

Tests were done on a 1.73 GHz double-core CPU playing back a CD-quality stereo audio file through the equalizer. The optimization step of downhill motion and tournaments as described in Section 2 is accomplished six times a second; the multiscale analysis is done twice a second. Without the heavy graphics, see Figure 2, the optimization step is sped up by approximately one third. Note that in a multitrack environment only one filter needs to be changed at a time so that the computation required for the optimization would not increase.

Thanks to the random mutations of every genotype, small changes in the user's input are immediately followed by appropriate changes in the best settings, independent of the mode. Gross changes in the input such as adding a new anchor point in mode 1 or redrawing half the curve in mode 3 require about one second until the optimization process settles. In all modes with exception of mode 1, this first result tends be replaced by a less complex (modes 2, 3, and 5) or better fitting (modes 2 through and 5) genotype two or three seconds later. Further huge improvements or drastic changes in the settings occur only rarely.

The multiscale computation consumes only little computational power: Without it, the number of iterations per second will increase by one tenth. To test in how far the multiscale analysis helps with the optimization, we used two user inputs for mode 3: first, a curve that resembles one period of a perfect sine wave with a range of -10 to 10 dB; second, a freehand curve with smaller features and steep cliffs. We ran four experiments in which each of these curves was approximated with a population generated from scratch: two runs in which new genotypes were created using the multiscale analysis and two runs that used random numbers instead. The results show that the multiscale analysis yields much faster progress, see Figures 9 and 10. (For easier reading, only the points of time where the best genotype is replaced are drawn.)

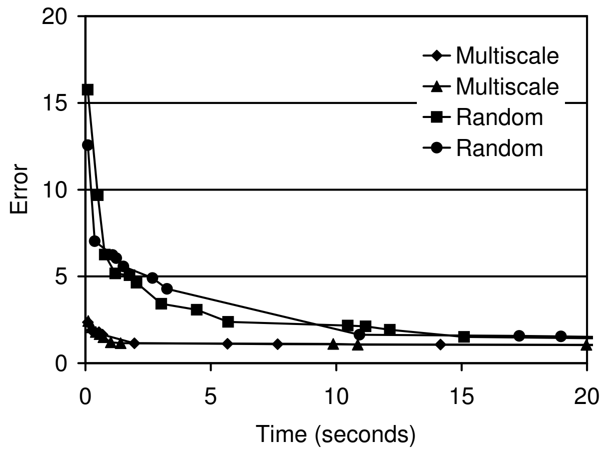


Fig. 9: A curve representing one period of a sine is easy to decompose for the multiscale algorithm. The residual error 1 is due to the complexity reduction mechanism, see Subsection 3.3.

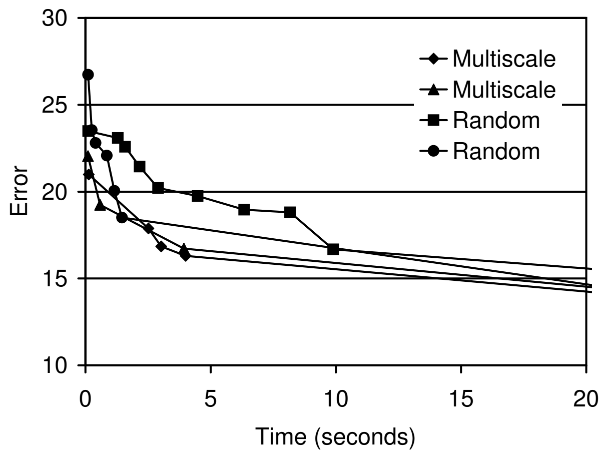


Fig. 10: A freehand curve with small as well as steep features poses more difficulties to the multiscale algorithm. A random initialization can be lucky to come close, see the series of data marked by circular dots.

At all times, the best genotype is used for real-time audio processing. Thus, changes in or the replacement of the best genotype may lead to annoying glitches in the output. However, most changes perturb the settings of the best current genotype by a small amount; replacements of the best genotype occur relatively rarely, only about five times for a large change in the user's input.

Some issues become evident during the use of the prototype:

- Mode 1: It is not very intuitive to place the anchor point that controls a shelving filter in the *middle* of its slope, see the rightmost control dot in Figure 3. This definition of the anchor point stems from the standard frequency controls.
- Mode 2: The optimization aims at making the curve pass virtually exactly through the given anchor points. This may lead to unexpected results. Hoping to get a shelving filter, for instance, one may place two dots at close frequencies but with a huge level difference. So steep a curve, however, cannot be realized by a shelving filter so that the optimization will produce a peak filter or even a combination of several filter bands.
- Modes 3 through 5: Drawing a curve turns out to be of limited value to tweak the settings. Whereas turning a knob results in continuous audio feedback, a curve has to be redrawn iteratively. The best use of these modes may be to do a small number of initial freehand sketches and then turn the optimization off, reverting to the standard controls.

5. CONCLUSION AND OUTLOOK

This paper demonstrated that an optimization process can be used to interactively control standard parametric equalizers. Modes 1 and 2 are particularly promising as they let the user continuously tweak the results. Mode 1, however, may be improved by defining the anchor point at the lower end of the "shelf" for a low-shelving filter and at the higher end for a high-shelving filter.

Even though late big improvements of the optimization and hence delayed large leaps in the parameter values are rare, they may occur. In order to prevent sudden and hard-to-localize maladjustments, the optimization could be stopped automatically after some time.

Future work may also address the audio glitches that appear when the best genotype's parameters change or the best genotype is replaced by another one. If only its parameters *change*, one can smoothen the change over time with no risk of instable combinations, as the parameters are those of a parametric equalizer, not the coefficients a_i and b_i of the filter polynomial. *Replacing* the filter without a glitch is more difficult, however. Running the old and the new version in parallel and cross-fading between them would be an option, but may lead to unwanted intermediate results due to phase cancellation. Another option may be to fill the delay elements of the new filter with values that ensure that the output of the new filter and its derivative equal those of the old filter when the switch occurs.

To speed up the optimization process greatly and to leave more CPU power for audio processing, the optimization could be handled mostly on the graphics chip [8]. This also leverages much of the graphics processing unit's potential, which still is heavily underutilized in audio software.

6. ACKNOWLEDGEMENT

This research was partly funded by grant 1742B04 of the German Federal Ministry of Education and Research (BMBF). The views and conclusions contained in this document, however, are those of the author.

7. REFERENCES

[1] G. Ramos and J. López, "Filter design method for loudspeaker equalization based on IIR parametric filters," J. Audio Eng. Soc. **54** (2006), no. 12, 1162–1178.

[2] A. Rimell and M. Hawksford, "The application of genetic algorithms to digital audio filters," presented at the AES 98th convention, Paris, France, 1995 February 25–28.

[3] K. Uesaka and M. Kawamata, "Evolutionary synthesis of digital filter structures using genetic programming," IEEE Transactions on Circuits and Systems II: Analog and Digital Signal Processing **50** (2003), no. 12, 977–783.

[4] V. Aggarwal and W. Jin, "Filter approximation using explicit time and frequency domain specifications," GECCO'06, 753–760.

[5] J.-T. Tsai, J.-H. Chou, and T.-K. Liu, "Optimal design of digital IIR filters by using hybrid Taguchi genetic algorithm," IEEE Transactions on Industrial Electronics **53** (2006), no. 3, 867–879.

[6] Y. Yu and Y. Xinjie, "Cooperative coevolutionary genetic algorithm for digital IIR filter design," IEEE Transactions on Industrial Electronics **54** (2007), no. 3, 1311–1318.

[7] R. Bristow-Johnson, "Cookbook formulae for audio EQ biquad filter coefficients," posted to the music-dsp list (2001).

[8] S. Harding and W. Banzhaf, "Fast genetic programming on GPUs," EuroGP 2007, 90–101.