Volume 55, Number 1, 2014

ACTA TECHNICA NAPOCENSIS
Electronics and Telecommunications

LABVIEW TOOL FOR DESIGNING FIR FILTERS FOR ACOUSTIC EQUALIZATION

Erwin SZOPOS[1] Călin FĂRCAŞ[2] Marina ŢOPA[1] Marius NEAG[1] Ioana SĂRĂCUŢ[1]
[1]Technical University from Cluj-Napoca, George Baritiu Street, Cluj-Napoca, 400027, Romania
[2]Radio Romania Cluj, Donath Street, Cluj-Napoca, 400331, Romania
Tel: +40264401803; Fax: +40264591340, erwin.szopos@bel.utcluj.ro

Abstract: The paper presents a tool for optimal synthesis of FIR filters tailored for the equalization of a room acoustic transfer function. The synthesis is optimized for minimizing the hardware resources required to implement the resulting filters, in particular by reducing the length of these filters. The optimization procedure employs a genetic algorithm developed for this application. The LabVIEW implementation of the filter synthesis tool is presented in some detail, along with a real-life example.

Keywords: *FIR filters, arbitrary magnitude response, acoustic equalization, optimization, genetic algorithm.*

I. INTRODUCTION

Digital filters are one of the most used systems in digital signal processing. Finite impulse response (FIR) filters have attractive properties: the stability can be guaranteed and linear phase can be easily achieved [1]. Therefore, they are popular in many applications such as communication systems, audio signal processing, biomedical instruments and so on. Unfortunately, standard synthesis procedures often yield FIR filters with a larger number of taps – thus more expensive to implement – than the infinite impulse response (IIR) filters which meet the same specifications.

Most of the numerous methods for synthesizing digital filters proposed in the literature deal with the case of having the wanted/imposed magnitude-frequency points (called hereafter the reference samples) uniformly distributed in frequency. However, there are quite a few applications which require reference samples placed non-uniformly on the frequency axis – for example, filters designed for hearing aids, where the reference samples are usually placed an octave apart [2], [3]. For such applications the standard methods can yield sub-optimal results. For example, when the standard version of the frequency sampling synthesis method [4] is applied to non-uniformly distributed samples one can obtain filters with very large ripple of the magnitude characteristic and very large spread of the filter coefficients, which in turn makes their implementation more difficult [5].

Several synthesis methods for FIR filters have been reported in the literature that take into account implementation requirements, aiming to provide FIR filters that not only accurately approximate the desired frequency characteristics, but are also effective for hardware implementations [6], [7].

This paper presents such a synthesis method, in fact a development of the improved version of the frequency sampling synthesis method proposed in [3] for FIR filters able to model human audiograms obtained by standard measurements, that yield sets of reference samples non-uniformly distributed in frequency. The main idea there was to derive new sets of target samples for the synthesis starting from the reference set, by changing the frequency position of the samples and adjusting their magnitude through interpolation; when necessary, additional samples were introduced. Moreover, the synthesis process in [5] involved an iterative optimization loop, with the aim of improving the matching between the resulting magnitude response and the desired one, while minimizing the filter length.

Here, the envisaged application is the equalization of the frequency response of an acoustic enclosure. Also, the optimization loop is driven by a genetic algorithm developed for this application. The design tool presented here was implemented in LabVIEW as it provides fast and efficient signal processing and the parameters can be easily adjusted on-the-fly.

The paper is organized as follows: Section II presents briefly the theoretical background of the standard frequency sampling algorithm followed by a brief description of the improved version proposed in [5]. Section III presents the strategy of the genetic algorithm that controls the optimization method while some details of its implementation by using LabVIEW are given in Section IV. A real life example is presented in Section V in order to demonstrate the validity of the proposed design tool. Conclusions are drawn in the last Section of the paper.

II. FIR FILTER SYNTHESIS

One of the standard methods for designing FIR filters is the frequency sampling algorithm [4]. FIR filters are classified in four types each with its own design equation. The discussion here focuses on the Type I filter (eq. (1)) because it fits most real-life applications. The synthesis for type 2, 3 and 4 FIR filters is similar [5].

$$V\cdot\left(h(0)\ \cdots\ h\left(\frac{N-1}{2}\right)\right)^T=\left(A_d(\omega_0)\ \cdots\ A_d(\omega_k)\right)^T \qquad (1)$$

Manuscript received November 1, 2013; revised January 30, 2014

The matrix V is computed as follows:

$$V = \begin{pmatrix} cos(\omega_0) & cos(2\omega_0) & .... & cos\left(\frac{N-1}{2}\omega_0\right) & 1 \\ cos(\omega_1) & cos(2\omega_1) & .... & cos\left(\frac{N-1}{2}\omega_1\right) & 1 \\ \vdots & \vdots & .... & \vdots & \vdots \\ cos(\omega_k) & cos(2\omega_k) & .... & cos\left(\frac{N-1}{2}\omega_k\right) & 1 \end{pmatrix} \quad (2)$$

where $A_d(\omega_k)$ is the desired magnitude frequency response, $h(n)$ represents the filter tap (coefficient) values and N is the filter length.

A major drawback of the frequency sampling method is that if the reference samples are distributed unevenly (in frequency), the taps of the resulting filter will be widely spread and this is not practical for implementation. In order to reduce the spread of the filter tap values the solution of repositioning the samples was adopted: Figure 1 shows how the initial frequency samples, i.e. $\{f_1, f_2, \dots f_k, \dots f_j\}$, are repositioned obtaining a new set of samples $\{f_1, f_{2,1}, \dots f_{k,1}, \dots f_j\}$ – called hereafter the design set. This approach has the advantage of maintaining a reduced filter length and a reduced spread for the filter taps, thus facilitating the implementation of the designed filter.

An interesting solution for optimum repositioning of the samples on the frequency axis is to use a genetic algorithm (GA) developed for this task. First, the GA generates randomly a design set representing the initial group of possible solutions to the design problem. For each design set a FIR filter is computed (eq. (1)), then evaluated and finally is ranked. Next the solutions are sorted according to their own rank and then the performance of the resulting filter both in time and frequency domains are compared against the maximum acceptable values set by the user. The cycle is repeated until a filter that meets all performance metrics is found or the maximum number of iterations (set by the user) is reached. If a filter that meets all the conditions is found, then another iterative cycle begins with the minimization of the filter length that is performed by deleting a sample from the current set. Next, the new set of samples is evaluated and if the new filter with reduced length also fulfills all the conditions then this cycle is repeated until no filter that meets conditions is found or the maximum number of iterations is reached. The performance metrics used in this paper are the spread of tap values – represented by the ratio $|h|_{max}/|h|_{min}$ – and the *sample-by-sample error* (*SSE*) - the distance between the desired and synthesized magnitude characteristics computed at the reference frequencies [5].

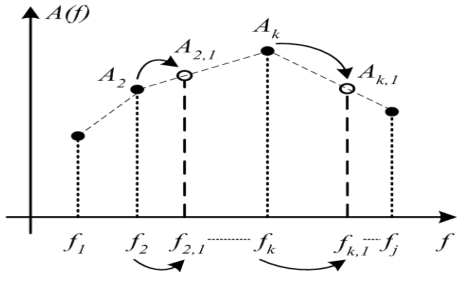

Figure 1. Obtaining the design set of samples, placed at different frequency points than the reference samples.

III. A GENETIC ALGORITHM DEVELOPED FOR THE OPTIMISED DESIGN OF FIR FILTERS

The GA emulates the processes of natural evolution based on the survival of the fittest individual within a population. The members of the population (the individuals) are represented by *chromosomes* that comprise several *genes* which can be modified through natural evolution phenomena such as crossover and mutation. The quality of each individual is given by a mark, called *fitness*, which is a weighted sum of several *objective functions* which express the various possible deviations of the solution from the ideal. A standard optimization algorithm based on GA consists of the following iterative steps [8], [9]:

- the algorithm starts with a “first guess” or a randomly generated group of chromosomes, that represent possible solutions for the problem being analyzed.
- each chromosome is evaluated by using a user-defined objective function. The resulting fitness score allows the ranking and sorting of chromosomes. In general, only the fit individuals, that is, those with good fitness marks, are allowed to participate to the creation of the next generation
- the next generation is obtained by emulating the evolutionary phenomena of mutation and crossover.
- the evolutionary cycle is repeated until the stopping condition is reached, that is either an optimum solution was obtained or the process has run for a set number of cycles.

Here, the chromosomes represent possible filter solutions; their genes encode the frequency-magnitude coordinates of the design set of samples, as shown in Figure 2 – note that the first and last sample remain unchanged so that all resulting magnitude characteristics cover exactly the same frequency band. The magnitude values are derived from the reference set of samples through interpolation.

This approach to encoding the filter parameters is more efficient than the straightforward way, which uses directly the filter coefficients [9], because the frequency position has always positive values (so there is no need to reserve a bit to represent the sign) and are usually large integers which do not require separate encoding of the integer and decimal parts. Moreover, it is not necessary to encode the entire frequency value: the position in frequency of all the samples except the first can be defined by their distance from the first sample, as the later position does not change. In most cases *6-8* bits are sufficient to represent the genes shown in Figure 2.

The objective function, F, used in this paper was chosen after trying out several options. It has the expression:

$$F = \sum_{k=0}^{M-1}\left(1 - \frac{A_{synth}(f_k)}{A_d(f_k)}\right)^2 \quad (3)$$

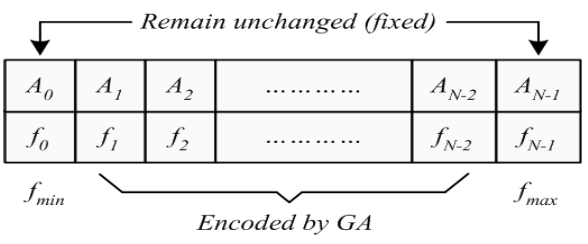

Figure 2. Structure of a chromosome, with genes that encode the frequency position of the design samples.

where M is the extended number of samples including the reference set and the intermediary points of the reference or the design sets, $A_d(f_k)$ and $A_{synth}(f_k)$ are the values of the desired and synthesized magnitude frequency characteristics.

IV. LABVIEW IMPLEMENTATION OF THE BUILDING BLOCKS

Figure 3 shows the flowchart of the virtual instrument (*vi*) used to generate the initial population that also includes the *check-and-delete clone* subroutine. In the first step the number of clones (parameter *NC*) and the index *i* of the main while loop are both set to *0*. In the second step, the control input *Init.Pop*, set by the user, decides which of the two modes will be selected: (a) the generation of an initial population with clone replacement or (b) only clone replacement in a given population. The population size *PS* and the number of bits *n* representing a gene are also set by the user at the beginning.

If *Init.Pop* is set to *TRUE* – mode (a) – then a new generation of individuals starts. This mode sets in the first iteration of the main loop ($i=0$) the maximum limit *U* of the outer for loop (represented by index *j*) to *PS*. In step three, a single individual will be obtained with the inner for loop (represented by index *k*). First, a random number corresponding to a gene is generated (*string* data type). Second, the current string as a gene is concatenated with the previous one and this way the loop is repeated *NDV* times

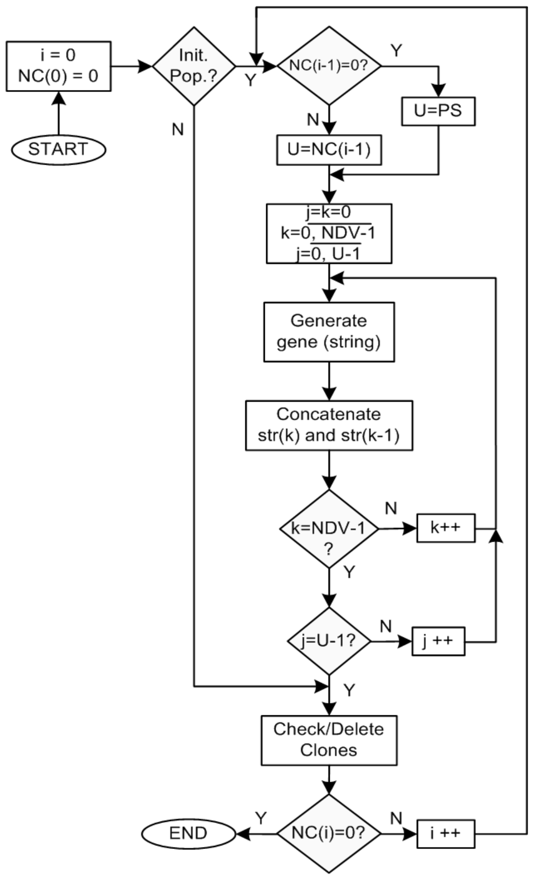

Figure 3. The flowchart of the program that generates the initial population and replaces the clones.

(where *NDV* is the number of design variables) thus obtaining a complete individual (chromosome). The string data type representing the individuals is chosen due to the possibility to extend the length of the individuals from 64 bits (maximum for the numerical data types) to a very large string of characters.

In step four, the outer for loop is responsible for collecting and putting the new individuals (chromosomes), generated by the inner for loop, into an array obtaining the new population. When the generation of the whole population is finished ($j_{max}=PS-1$) the clone elimination and replacement subroutine starts (Figure 4). If $NC\neq0$ (at least one clone exists) then the main loop (steps three and four) that generates new individuals is repeated but with the maximum limit *U* for the outer for loop set to *NC(i-1)*. In this subroutine, first, the existing clones are deleted from the population and then a number of new individuals equal to *NC(i-1)* is generated and included at the end of the population array. The clone elimination and replacement subroutine runs until $NC=0$ providing the new population. If *Init.Pop* is set to *FALSE* – mode (b) – then the execution of this *vi* starts with the subroutine to replace clones in a given population, described above. Figure 4 shows the *vi* for identifying and deleting the clones in a given population (*Pop_In*). In the first iteration, this subroutine compares the element from index *i* of the array *Pop_In* with the rest of its elements starting from index $i+1$. If there are equal elements with it then all of them will be deleted iteratively in the inner while loop. Each deletion is counted and stored in memory through a shift register representing the number of clones *NC(i)* from the current iteration *i*. The number of clones *NC(i)* is then used, in the next iteration $k+1$, to compute the next element's index to be analyzed.

After the clones were deleted the two while loop stop if there is no other element in the *Pop_Out* array to be analyzed. This *vi* provides at the output as well the total number of clones *NC* that is the sum of the number of clones per outer iteration.

Each individual from every population is tested to see how good it is as an optimal numerical solution for the proposed problem. Before the fitness score computation (3) the information (design variable) encoded by the GA through the individuals has to be decoded. In our application the reference/design set samples positions (f_k) are represented by genes. The *vi* for the gene decoding is shown in Figure 5. In order to decode a gene one needs first to compute its length, by dividing the length of an individual by the number of design values (*NDV*). Next, the genes of each individual are extracted one-by-one with the *for loop*; here, the clones between the genes are also replaced with other non-clone genes using the algorithm from Figure 3 (*Init.Pop=FALSE*, *NDV=1*) and finally the genes values, represented with string data type, are converted into the integer type (*Gene_Dec*), thus, prepared for numerical computations.

Figure 6 shows how the numerical values of the genes contribute to the computation of the design variables values f_k. This *vi* is the practical implementation of the expression:

$$f_k = f_{min} + \frac{f_{max} - f_{min}}{2^n + 1} \cdot \left(g_k + 1 \right) \tag{4}$$

where f_{min} and f_{max} are the minimum and the maximum frequencies in the reference set; n is the number of bits representing a gene; g_k is the numerical value of a gene at iteration k. For each of the genes there is a frequency corresponding value f_k computed by (4); these values are sorted in ascending order and put into a vector providing the new positions of the reference set samples.

With the new design set obtained above a FIR filter is designed using (1) then the resulting filter taps $h(n)$ and the corresponding magnitude response $A_{synth}(f_k)$ are evaluated, receiving, after evaluation of each individual, a score determined by (3).

Once all the individuals from a population are evaluated they are sorted in ascending order according to their score. Next, a number of individuals (equal to *PS*) considered being the best fitted ones are returned and sent to the reproduction block where the input parameters set by the user are: *PS*, *NDV*, the number of bits *n* representing a gene and *PM* (probability of mutation). This block performs the genetic operators such as: the selection, the crossover and the mutation. In this work a particular method for selection is implemented where every individual from a population has the safe chance to breed. The operating stages of this block are: select the first two individuals (parents) from *Pop_In*, crossover of the individuals, mutation, build offspring array and replace the possible clones. The crossover operation is performed in one point by generating a random index and then all data before that point in two individual's strings are swapped. After the crossover operation two offspring are obtained and on a single bit on each of them the mutation operation is applied with a specific chance of probability (*PM*). In this way the number of the offsprings will be the same as the number of the parents, thus, having control over the resulting population size. This particular selection method was compared to the well known *roulette wheel* selection method but no significant differences were observed in operating, the advantage being its simplicity of implementation.

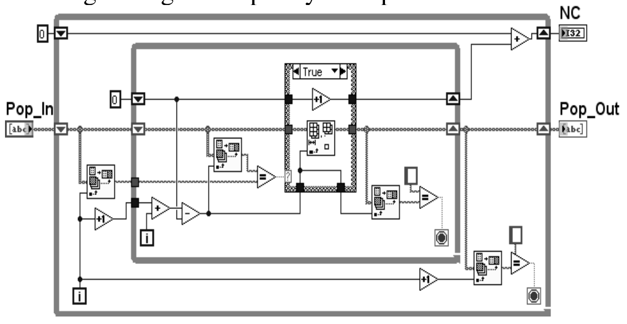

Figure 4. The vi for clone detection and deletion.

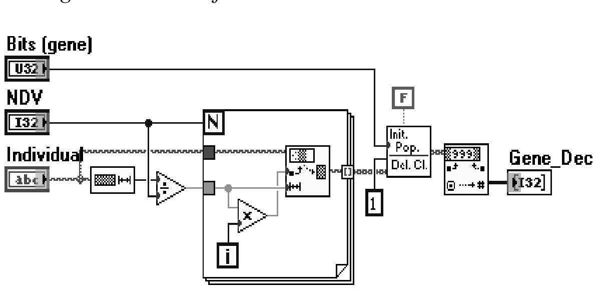

Figure 5. The vi for gene decoding.

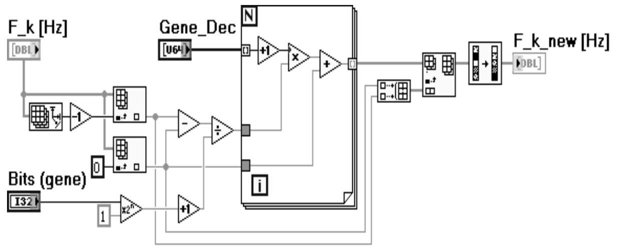

Figure 6. The vi for new design variables f_k computation.

V. VALIDATION OF THE SYNTHESIS METHOD

In order to validate the efficiency of the synthesis tool described here, this section presents a real life application: the design of the FIR filter necessary to implement a room response equalization system. This filter has to compensate the room transfer function (*RTF*) of the acoustic enclosure, which characterizes the path from the sound reproduction system to a listener within the room [10].

A. Desired magnitude characteristic of the equalizing filter

The first step in designing a response equalization system is to determine the *RTF* of the acoustic enclosure. There are several methods for deriving the *RTF* through experimental measurements [11]. The main distinction is made between single- and multiple-position (of the listener) measurements; in our case measurements have been performed in five positions within the room, then an average was calculated. The bandwidth of the available sound equipment, *63 Hz-12.5 kHz*, was split into *25* equal-in-*dB* sub-bands, each covering a third of an octave.

Figure 7 presents the layout of the room; the area of interest is the middle section, delimitated by the four signal sources *S1-S4*; *P1-P5* indicate the position of the five measurement points.

Figure 8 shows the distribution of the values for the acoustic parameter *Clarity 80* (*C80*) [12], measured in the sub-band centered on *1 kHz*, for the five positions, *P1-P5*. Obviously, there are significant differences in the sound quality between these points; in order to characterize the entire room one has to use some averaging, selecting the type that suits best the application. The usual selection criteria are the values of the standard deviation and the variance – the larger, the better [13].

Table 1 summarizes the standard deviation and the variance values obtained for the main averaging options: root-mean-square (*RMS*), mean, median and Min-Max. It is clear that the Min-Max average is the best choice here:

$$\left|H_{mm}\left(e^{j\omega}\right)\right| = \frac{1}{2}\left(\max_k \left|\hat{H}_k\left(e^{j\omega}\right)\right| - \min_k \left|\hat{H}_k\left(e^{j\omega}\right)\right|\right) \quad (5)$$

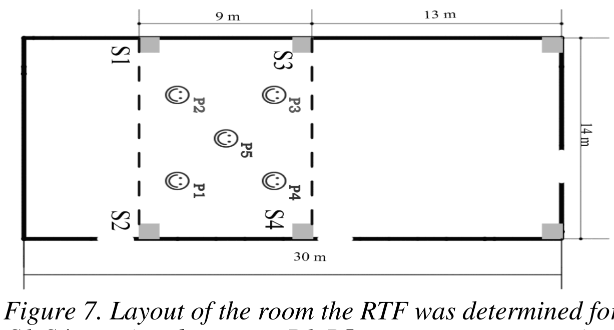

Figure 7. Layout of the room the RTF was determined for. S1-S4 are signal sources, P1-P5 are measurement points.

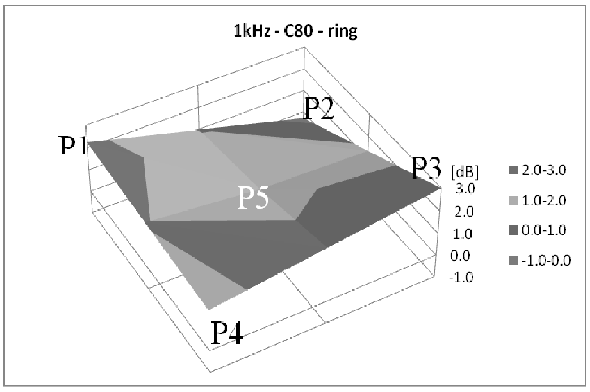

Figure 8. Distribution of the clarity values C80 measured in the 1 kHz sub-band at the locations P1-P5 in Figure 7.

Table 1. The values of standard deviation and variance for four types of averaging performed on measurements results obtained in points P1-P5.

Average Type	Standard Deviation	Variance
RMS	*5.600386*	*31.36433*
MEAN	*5.542928*	*30.72406*
MEDIAN	*5.249329*	*27.55545*
MIN-MAX	*5.989202*	*35.87054*

These operations were performed for each of the *25* frequency sub-bands; the results are represented in Figure 9 by the squares on the interrupted line; the *RTF* resulted by interpolating between these points, that is the magnitude characteristic drawn with interrupted line in Fig. 9.

Once the *RTF* is known one can derive the desired magnitude characteristic of the equalizing filter; several methods have been propose in the literature, such as the least-square approach in [14]. Here we used a simpler approach: the mirror image of the *RTF* with respect to a reference level, set here to *36 dB* after several experiments regarding the subjective quality of sound. The resulting desired characteristic for the equalizer was drawn in Fig. 9 with continuous line; the squares on top of that line indicate the reference set of samples for the equalizing filter.

B. Optimized equalizers designed with the proposed tool

Obviously, the main design requirement for the equalizing filter is to closely match the desired magnitude characteristic shown in Figure 10. In this case the *maximum acceptable magnitude error* (*MME*) - the maximum distance allowed between the synthesized and desired magnitude characteristics – was set to *5 dB*. This value matches the resolution of the measurements the *RTF* was obtained from.

The minimization of the filter length was imposed as an optimization criterion. Therefore, as a first design iteration, the synthesis starts with a filter with the same number of taps as the one produced by the standard frequency-sampling method, that is *49* in this case (there are *25* samples in the reference set). Once a solution is found the number of taps is decreased by a unity and the next design iteration starts. Thus the number of taps will be decreased until a minimum is reached, a further reduction of which makes the synthesis tool unable to find a valid solution that is the resulting filters no longer meet the *MME* requirement. In this case the smallest number of taps we obtained solutions for was *25*.

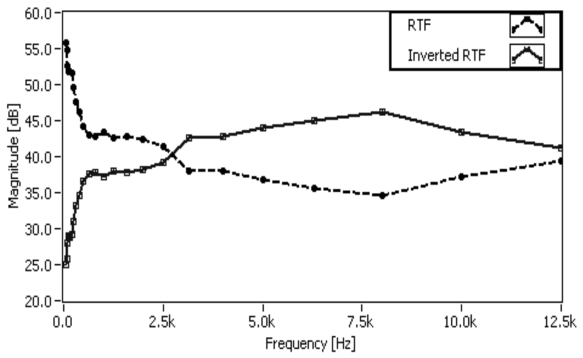

Figure 9. The room transfer characteristic (dotted line) and the desired magnitude characteristic of the equalizing filter (continuous line)

The following parameter values were used to set up the GA for this application: the number of bits for representing the design variables *n = 6*; the population size *PS = 100*; the number of design variables *NDV = 23* (the maximum value of *NDV* equals the number of reference samples minus *2*, as the first and last samples are maintained unchanged).The probability of mutation was set relatively high, *PM = 0.06,* in order to avoid stagnation at a suboptimal solution.

Figure 10 presents the magnitude characteristics for two valid solutions provided by the synthesis tool: a filter with *49* taps (continuous line) and one with *25* taps (dotted line).

As these characteristics have no significant ripple, the matching between them and the desired characteristic can be assessed by using the *SSE* metric described in Section II.

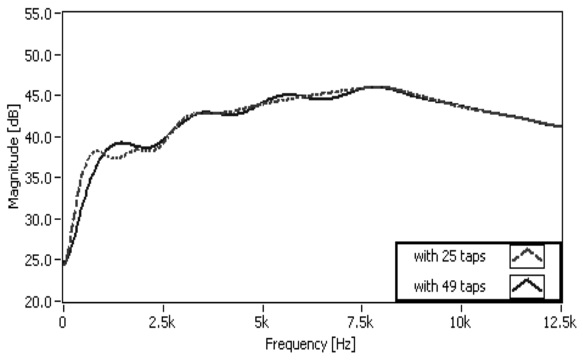

Figure 10. The magnitude characteristics of two synthesized equalizers: one with 49 taps (continuous line) and the other with 25 taps (dotted line).

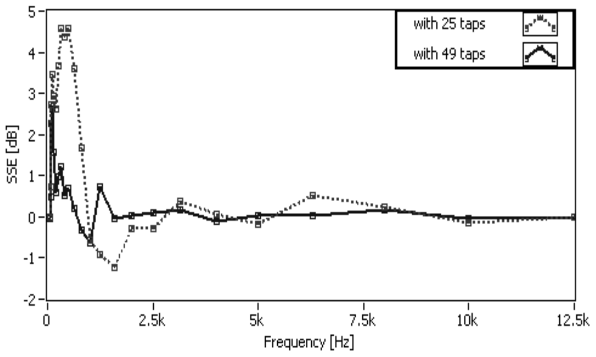

Figure 11. The sample-by-sample error for the 49-taps (continuous line) and 25-taps (dotted line) equalizers.

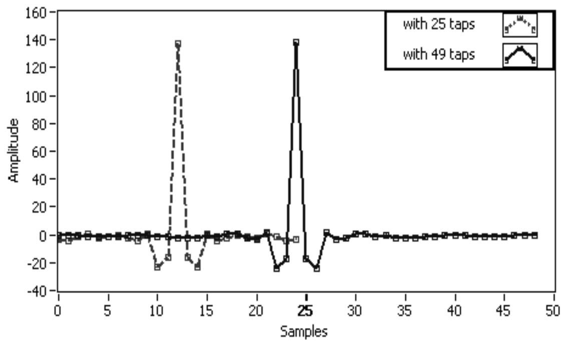

Figure 12. The impulse responses of the two synthesized equalizers: 49-taps continuous line, 25-taps dotted line.

Table 2. Main parameters of the 49- and 25-taps proposed equalizers, against the standard 49-tap filter

Parameter	Standard *49-taps*	Optimized *49-taps*	Optimized *25-taps*
MME[dB]	*203*	*2.5*	*4.5*
$\|h\|_{max}$	*2.8E+8*	*1.5*	*1.8*
$\|h\|_{min}$	*2.5E+11*	*138.49*	*137.24*
$\|h\|_{max}/\|h\|_{min}$	*892.86*	*92.3*	*76.24*

Figure 11 gives the *SSE* values for the *49*- and *25*-taps filters presented in Figure 11. As expected, the equalizer with *49* taps approximates better the desired magnitude characteristic than the *25*-taps filter, with the maximum *SSE* value below *2.5 dB*. However, it should be noted that, despite its much shorter length, the *25*-taps equalizer provides an acceptable solution, as its maximum *SSE* value is *4.5 dB*, and that for only one point of the characteristic. FIR filters with shorter length are preferred not only for ease of implementation but also for avoiding/reducing the overlapping echoes that can distort the processed signals.

Figure 12 presents the impulse responses of the two synthesized equalizers; both have smooth shapes, as predicted by the absence of significant ripple in their magnitude characteristics shown in Figure 10.

Table 2 summarizes the main parameters of the two optimized equalizers, compared against the *49*-taps filter obtained by using the standard frequency sampling method For the standard filter the *SSE* is nil for all the reference points but the ripple between these points is huge, leading to the *MME* values over *200 dB*! For the optimized filters the *MME* equals *SSE*, in both cases with values below the set limit of *5 dB*. Also, the spread of tap values is significantly smaller for these equalizers than for the standard filter.

VI. CONCLUSIONS

This paper presents a design tool developed for the synthesis of FIR filters that closely match a desired magnitude characteristic that can be defined by frequency-magnitude points non-uniformly distributed in frequency. The envisage application is the equalization system that compensates for the transfer function of an acoustic enclosure, which characterizes the path from the sound reproduction system to a listener within the room.

The design engine at the core of this system is the frequency sampling method. An iterative synthesis process was set up in order to overcome the limitations of this method when dealing with non-uniformly distributed samples: in each of design iteration a new set of target samples for the synthesis is derived, starting from the set used in the previous iteration, by changing the frequency position of the samples and adjusting their magnitude.

The synthesis is organized as an optimization design loop, driven by a genetic algorithm developed for this application that provides the target samples for each design iteration. The GA takes into account not only the main requirement regarding the magnitude characteristic but also requirements related to the ease-of-implementation of the resulting filters, particularly the minimization of their length. The synthesis procedure was detailed only for type I FIR filters (odd length, even symmetry) but it can be extended to type 2-4 filters.

The effectiveness of the proposed synthesis tool was demonstrated by a real-life example: the design of FIR filters able to equalize the acoustic transfer function of a room. The procedure employed for deriving the room transfer function is presented in some detail; from there, the design target was derived – the desired characteristic of the equalizing filter, defined by 25 frequency-amplitude points.

The synthesis tool provided a number of valid solutions to this problem, filters that closely match the desired characteristic, with lengths varying from 49 taps (same as the filter obtained by applying the frequency sampling method in the standard way) to only 25 taps.

REFERENCES

[1] D. Schlichtharle, "*Digital filters. Basics and design*", Springer, 2000

[2] P. Srisangngam, S. Chivapreecha, K. Dejhan, "A design of IIR based digital hearing aids using genetic algorithm", *8th Int. Conf. on Electrical Engineering, Electronics, Computer, Telecomms and IT,* 2011, pp. 967-970

[3] E. Szopos, M. Topa, L. Festila, H. Hedesiu, "FIR Synthesis of The Human Hearing Mechanism Response", *Acta Technica Napocensis – Electronics and Telecomunications*, 2010, 51, (4), pp.41- 44

[4] T. Parks, C. Burrus, "*Digital filter design*", John Wiley & Sons 1987

[5] E. Szopos, M. Neag, I. Saracut, H. Hedesiu, L. Festila, "A method for designing FIR filters with arbitrary magnitude characteristic used for modeling human audiogram", *Advances in Electrical and Computer Engineering*, 2012, 12, (2), pp. 51-56

[6] D. Maskell, "Design of efficient multiplierless FIR filters" *IET Circuits Devices Syst.*, 1, (2), 2007, pp.175-180

[7] R. Mahesh, A. Vinod, "Low complexity flexible filter banks for uniform and non-uniform channelisation in software radios using coefficient decimation", *IET Circuits Devices Syst.*, 2011, 5, (3), pp. 232–242

[8] D. Goldberg, "*Genetic algorithms in search, optimization and machine learning*", Addison-Wesley, Longman Publishing Co., 1989

[9] S. Ahmed, "Design of FIR filters with arbitrary amplitude and phase specifications using genetic algorithm", *IEEE 46th Midwest Symposium Circuits and Sytems*, 2003, 2, pp. 648-651

[10] A. Carini, S. Cecchi, F. Piazza, I. Omiciuolo, G. Sicuranza, "Multiple Position Room Response Equalization in Frequency Domain", *IEEE Transactions on Audio, Speech and Language Processing*, Vol. 20, No. 1, 2012, pp. 122-134.

[11] A. Farcas, M. Topa, "On The Choice Of The Method For Obtaining The Room Impulse Response", Acta Technica

Napocensis, Electronics and Telecommunication, Vol. 54, Nr. 1, 2013, pp. 34-42

[12] M. Topa, N. Toma, B. Kirei, I. Saracut, A. Farina, “Experimental Acoustic Evaluation of an Auditorium”, Advances in Acoustics and Vibration, Article ID 868247, 2012.

[13] Y. Haneda, S. Makino, Y. Kaneda, “Multiple-Point Equalization of Room Transfer Functions by Using Common Acoustical Poles”, IEEE Trans. on Speech and Audio Proc., Vol. 5, Nr.. 4, 1997, pp. 325-333.

[14] J. Mourjopoulos, P. Clarkson, J. Hammond, “A comparative study of least-squares and homomorphic techniques for the inversion of mixed phase signals,” *Proc. ICASSP 1982, Int. Conf. Acoust., Speech, Signal Process.*, 1982, pp. 1858–1861