ARCHIVES OF ACOUSTICS
29, 2, 177–189 (2004)

GENETIC ALGORITHM-BASED ACTIVE NOISE CONTROL SYSTEMS – SIMULATIONS USING INTERNET

G. MAKAREWICZ

Central Institute for Labour Protection
National Research Institute
Czerniakowska 16, 00-701 Warszawa, Poland
e-mail: grmak@ciop.pl

(*received 17 March 2004; accepted 26 March 2004*)

The application of genetic algorithm in Active Noise Control (ANC) systems is described here. Advantages and disadvantages of genetic algorithm-based ANC systems are discussed. Two types of genetic algorithm: binary and continuous parameter, in the context of ANC systems, are analyzed. The software for simulation of ANC systems based on gradient and genetic algorithms is described. The software consists of modules which represent functional elements of ANC systems. These modules can be easily implemented into programs for simulation of specific structures of ANC systems. The software was written in JAVA and simulations can be performed using Internet. Some results of the simulations are described. The software is currently available on www.anc.pl.

1. Introduction

The genetic algorithms are finding best solution algorithms utilizing well known natural selection mechanisms [1, 2, 9, 12]. The genetic algorithms feature high robustness, so that searched spaces are not affected by commonly occurring limitations, such as continuity, unimodality of the objective function, etc. Easiness of adjustment to different requirements and simplicity of its fulfilment were the reason why, upon development of useful procedures realising genetic algorithm functions, the genetic algorithms have become commonly used for optimization issues [2, 5].

Despite ongoing development of the genetic algorithm theory, some features of the genetic algorithm have not been recognized yet. It happens that developed mathematic foundations of the genetic algorithm prove true only in some applications, but in other applications, for unknown reasons, they do not prove [2]. Due to that, different non-standard solutions are used for individual applications, in which the genetic algorithm's parameters are adjusted experimentally, or even using the trial-and-error method [2, 9].

Simulations are thus a very important element related to the genetic algorithm utilization. The simulations enable selecting the most advantageous parameters for the genetic algorithm before the algorithm is used in the given solution. There is a variety of specialized software packages purposed for such a kind of research. However, these packages are very complex and usually offer no functions focused on an active noise control. The software described here may fill this gap. This software is easy to use, has been developed with active noise control in mind, and the most important thing, ready-to-use simulation programs may be shared via the Internet [7].

In this paper, some terms commonly used in connection with genetic algorithms for the life sciences have been replaced with corresponding terms used in active noise control systems. Thus: *the population* term is replaced with *the set*, *the individual* term is replaced with *the filter*, *the chromosome* term is replaced with vthe filter coefficient vector, and *the gene* is a *single coefficient of this filter*. Remaining terms (e.g. crossing, mutation, selection, etc.) remain unchanged.

2. Utilization of the genetic algorithm in active noise control systems

Most of currently used ANC systems are digital systems [3, 6]. The controller with a digital filter is responsible for correct synthesis of the compensation signal. This may be the simplest filter with finite impulse response (FIR), but also a complex neural network. The controller is responsible for such a selection of the filter coefficients, so as to obtain the highest noise reduction. In ANC systems, gradient algorithms are used for the filter coefficient adaptation. ANC system operates in cycles; in each cycle, based on an error signal, the filter coefficient vector's value is adjusted in order to gain the optimal form. The filter quality is indicated by the noise reduction level possible to gain.

The genetic algorithm may also be used to find the optimal digital filter form (the optimal coefficient vector) [4, 8, 10, 11]]. The set of n filters, represented in forms of the filter coefficient vectors (Fig. 1) must be created. Successive coefficient vectors are retrieved from the filter set; these coefficients are then inserted as the ANC system's controller coefficients (**i** vector in Fig. 1). For each filter, based on an error signal measurement (adder output in Fig. 1), its adaptation is determined (the lower the error signal is, the higher the filter adaptation). Once the entire set has been verified, all filters are assigned the values which decide on their quality and affect the probability of their copying in next generation.

Copying is related to the selection and crossing operations. The most popular selection method is the one where the filter selection's probability is proportional to its adaptation. New filter set is built via crossing and mutation, and the process is repeated. Due to the fact that the selection probability of the filter for crossing operation is higher when the error signal received in ANC system in which this filter is used is lower, in next generations the set is made up of filters with even better quality, i.e. the filters providing the better and better noise reduction.

The parameter coding method is a very important aspect affecting the genetic algorithm operation. The most commonly used classification is the division of algorithms

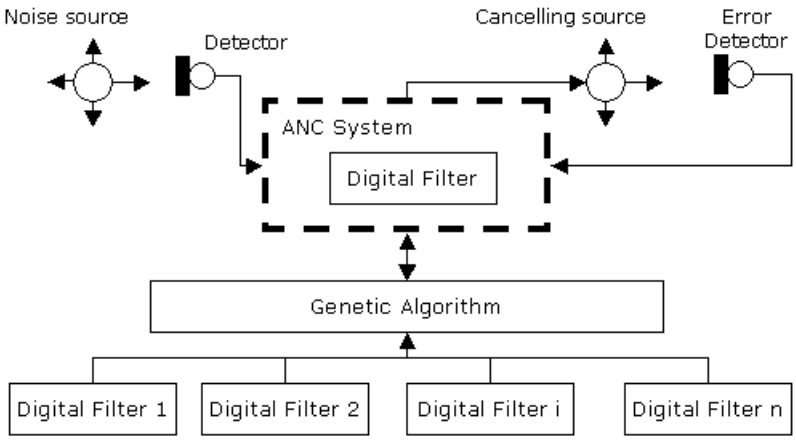

Fig. 1. Linking the genetic algorithm elements with ANC system.

into so called binary and continuous algorithms. For a digital filter used in ANC system it means that also filter coefficients are represented in binary or continuous/floating point form.

The parameter coding method selection significantly affects the crossing operation. For binary coding, the coefficient exchange between vectors ensures that the entire parameter space is searched, the size of which is limited only by the bit number in coding method. For continuous coding, application of the crossing operation in analogous way as for binary coding, usually brings no positive effects. First of all, due to the fact that a single vector element represents the entire parameter, a single-point crossing, frequently used in binary algorithms, is not sufficient. This is mainly due to the fact that for the same optimization task the continuous coefficient vector is much shorter than its binary equivalent, and the accuracy of parameter space searching when exchanging all parameter groups may be too low. The solution for this problem is utilization of multi-point crossing that ultimately may be transformed into the method in which all parameters of selected filter coefficient vectors are crossed.

The genetic algorithm principle prevents from using its adaptation features directly when operating ANC system. Thus, in each genetic algorithm-based ANC system, two operating phases are distinguished. The first phase is a period to find the most advantageous controller form. In this phase, ANC system performs the evolution functions against code series created based on digital filter coefficients. Determination on an objective function for each controller's form means that ANC system may both reduce and amplify the noise level. In the next generations the controller forms converge to the optimal form, but as a result of mutation, there is a definite (greater than zero) probability of generating the form of controller which amplifies the noise level. Therefore, the described operating phase may not be deemed as "normal" ANC system operation. If one of the controller forms meets assumed objective criterion and is deemed the optimal one, ANC system turns to the next operating phase. The controller parameters are saved

in controller's memory and the system operates as classical ANC system. In this phase, ANC system may both operate as constant parameter system or adaptation system.

3. Simulation software

A lot of currently available programming tools enable simulating via Internet. Java language has been selected as a programming language. This selection was due to Java's computing capabilities, its commonality and object programming capabilities. Figure 2 presents the structure of simulation software.

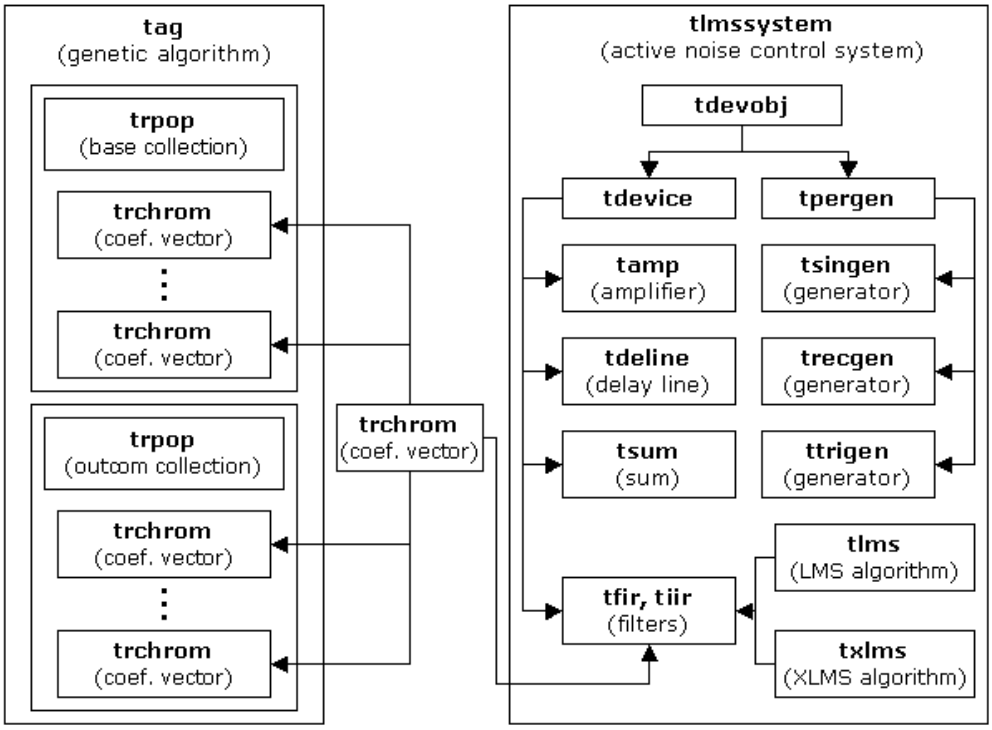

Fig. 2. Organization of class package designed for examination of the genetic algorithm usage in ANC systems.

The most important class used in this software is **trchrom** class, representing filter coefficient vector. For binary coding, the coefficient vector length equals the product of coefficient number and the number of bits for binary coding; for continuous coding, it equals the filter coefficient number. Figure 3 presents the structure of digital filter with finite impulse response (FIT) of L order, in which an input signal y_k is yielded as a result of operations against an input signal samples x_k:

$$y_k = \sum_{n=0}^{L} b_n x_{k-n} \tag{1}$$

and corresponding coefficient vectors with binary (4-bit coding) and continuous representation.

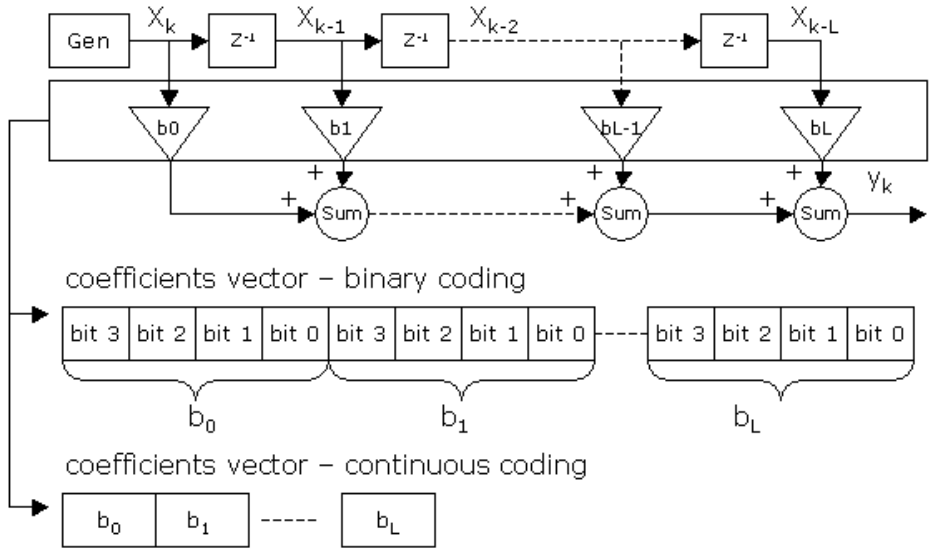

Fig. 3. Coefficient vectors of finite impulse response filter for binary and continuous coding.

The digital filter coefficient's b_i order in vector doesn't matter. The genetic algorithm is not order-sensitive.

The **trpop** class represents digital filter set. The genetic algorithm the functions of which are fulfilled by **tag** class requires storing two sets of filters, of which one contains the base (parent) filters, and the other contains target (descendant) filters being a result of the genetic algorithm operation against the base filters.

As mentioned above, the coding method for the filer coefficient values is a very important element related to the genetic algorithm. The continuous coding features another problem: in successive generations, the filter coefficient values are not changed, but only their positions are changed. Therefore, despite the theoretically continuous parameter space, during the genetic algorithm operation in ANC system we are working with a limited filter set. The best filter in this set may significantly vary from the optimal form. For more accurate searching of the parameter space a mutation may be used, but this occurs with a very small probability, thus relying on this mechanism has no practical meaning. Therefore, more complex crossing methods have been implemented in the software, to gain new parameter values. In the first method, b parameter values are determined as in the equation below:

$$\begin{aligned} b_{\text{descendant1}} &= \beta b_{\text{parent1}} + (1-\beta)\, b_{\text{parent2}}, \\ b_{\text{descendant2}} &= (1-\beta)\, b_{\text{parent1}} + \beta b_{\text{parent2}}, \qquad \beta = 0 \ldots 1. \end{aligned} \tag{2}$$

The limitation of this method is that gained parameter values always fall within the range limited by the values of the base filter set parameters. To enable generating off-range values, we may use the method as in the equation below:

$$\begin{aligned} b_{\text{descendant1}} &= 0.5 b_{\text{parent1}} + 0.5 b_{\text{parent2}}, \\ b_{\text{descendant2}} &= 1.5 b_{\text{parent1}} - 0.5 b_{\text{parent2}}, \\ b_{\text{descendant3}} &= -0.5 b_{\text{parent1}} + 1.5 b_{\text{parent2}}. \end{aligned} \tag{3}$$

As shown above, three resulting coefficients are created based on two base coefficients. One of them is rejected (its value not meeting assumptions), the other two are included in resulting filter coefficient vectors. The heuristic crossing method is also useful for parameter generating:

$$\begin{aligned} b_{\text{descendant1}} &= \beta_1 \left(b_{\text{parent1}} - b_{\text{parent2}}\right) + b_{\text{parent1}} \qquad \beta_1 = 0 \ldots 1, \\ b_{\text{descendant2}} &= \beta_2 \left(b_{\text{parent1}} - b_{\text{parent2}}\right) + b_{\text{parent1}} \qquad \beta_2 = 0 \ldots 1 \end{aligned} \tag{4}$$

and its variation, described with the equation as below:

$$\begin{aligned} b_{\text{descendant1}} &= b_{\text{parent1}} - \beta \left(b_{\text{parent1}} - b_{\text{parent2}}\right), \\ b_{\text{descendant2}} &= b_{\text{parent2}} + \beta \left(b_{\text{parent1}} - b_{\text{parent2}}\right) \end{aligned} \tag{5}$$

according to which, depending on β coefficient value, it is possible to generate resulting coefficient values, placed between base coefficient values ($\beta = 0...1$) or exceeding them ($\beta > 1$).

The genetic algorithm is used only for finding the optimal form of the digital filter. Thus, apart from the classes related to this algorithm, simulating required also development of the class hierarchy which can be used to create a model of ANC system. Moreover, the genetic algorithm functions had to be linked appropriately with ANC system functions. On the right of Fig. 2, the structure of the classes representing building blocks of ANC system is shown. The element linking the genetic algorithm with this system is previously described **trchrom** class – digital filter coefficient vector. Figure 4 presents detailed relations between several example classes included in the software, and building blocks (devices) of ANC system.

The base class for all devices is **tdevobj** class, representing single-output device with the parameter vector and the virtual signal processor, which, based on a given algorithm and parameter values, transmits the signal samples to device's output. Creating classes representing specified single-output device requires only determination of the parameter set and the function used for an output signal calculation. This is presented in Fig. 4 by **tpergen** and **tsingen** classes, representing periodic wave generator and sine wave generator, respectively. The expansion of **tdevobj** class is **tdevice** class which represents universal multi-input device. Based on this device (Fig. 2), the ANC system

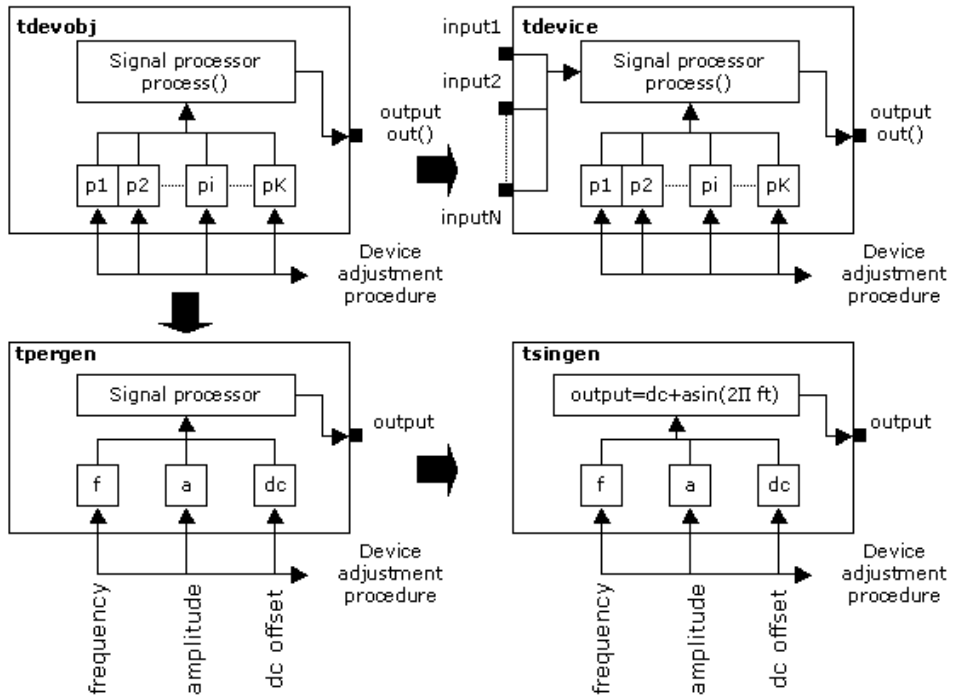

Fig. 4. The internal structure of **tdevobj, tdevice, tpergen** and **tsingen** classes along with the inheritance mapping.

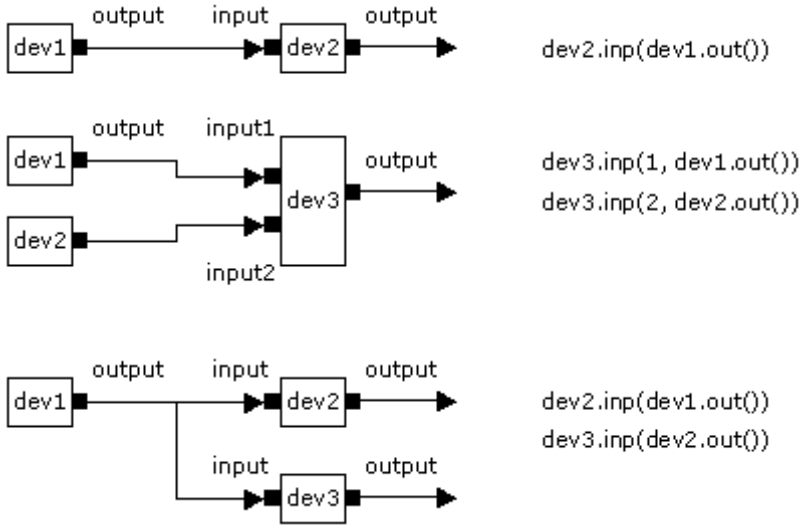

Fig. 5. Example connection lists.

building blocks are created, such as adders, amplifiers, delay lines, etc. The **tdevobj** and **tdevice** classes are equipped with inp() and out() methods, enabling device interconnection. Recalling them in respective order allows to map any complex structure of ANC system.

Figure 5 presents the example lists of connections between ANC system blocks.

The class hierarchy as in Fig. 2 is open. It is possible to add new classes to the existing ones, but also to create child classes that inherit properties of the classes implemented in the software.

4. Example simulation results

Using developed software, series of simulation calculations have been performed, and programs were developed enabling interactive simulations via Internet. Below, two software usage examples are presented.

Figure 6 presents an example of ANC system based on the genetic and least mean square (LMS) algorithm. The system is composed of six objects:

1) `gen` – the compensated signal generator, simulating noise source,
2) `del` – the delay line, simulating an acoustic path (e.g. wave-guide) between the noise source and the compensating source,
3) `sum` – the adder that corresponds to the location of the compensating source,
4) `fir` – the finite impulse response filter,
5) `amp` – the amplifier that controls the compensating source,
6) `lms` – the LMS algorithm used for the digital filter coefficient adaptation.

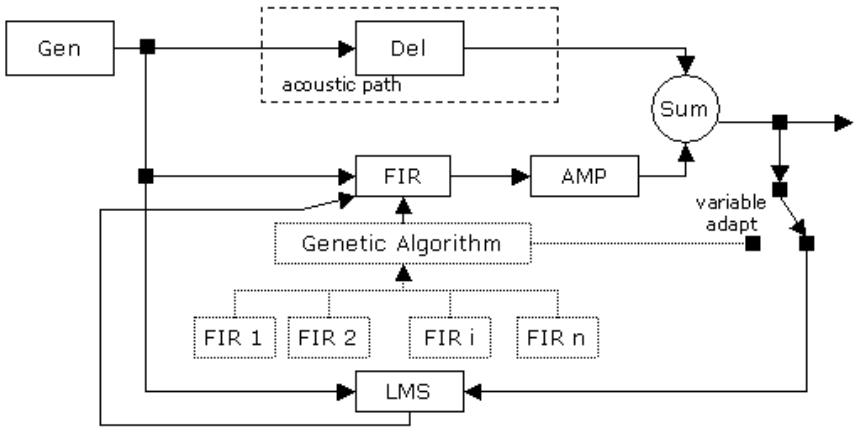

Fig. 6. ANC system structure.

According to the rules as mentioned before, the building block interconnection relies on recalling of inp() and out() methods in proper order. For example, transferring the samples from *Gen* generator to *Del* delay line input is fulfilled by the following function:

```
del.inp(gen.out())
```

and transferring the samples from *FIR* filter output to *Amp* amplifier input is fulfilled by the following function:

```
amp.inp(fir.out())
```

The interconnection list that maps the structure of the system in Fig. 6 is presented in the source code fragment in Table 1.

Table 1. Mapping of ANC system structure presented in Fig. 6.

```
del.inp(gen.out());
fir.inp(gen.out());
sum.inp(1, del.out());
amp.inp(fir.out());
sum.inp(2, amp.out());
if(adapt)
lms.process(sum.out());
```

Table 1 includes also the conditional instruction, which monitors a value of the logical variable "adapt" and enables/disables operation of the adaptation algorithm. Figure 7 presents example signal waves in the most important system points.

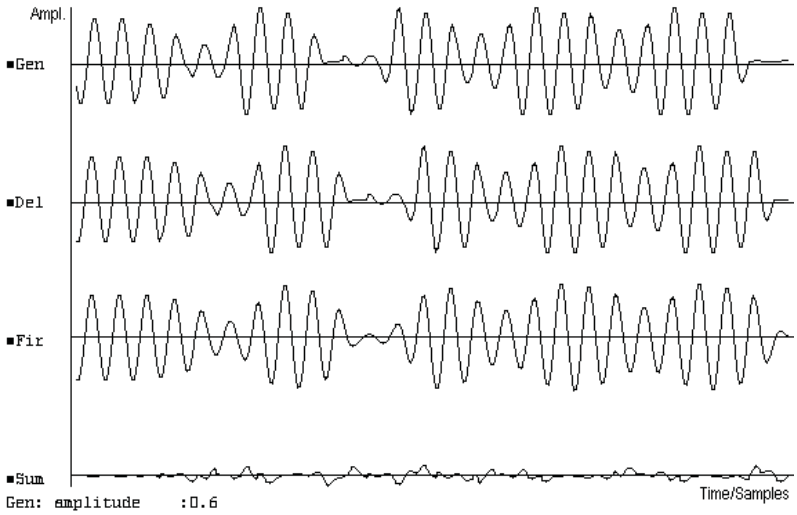

Fig. 7. The waves presenting the ANC system response to the interferences (the compensated signal amplitude's change).

The signal processing is started upon program launch, as in a real ANC system. The digital *Gen* generator transfers at specified frequency compensated signal samples, which are processed according to implemented ANC system structure (Fig. 6 – Table 1). Like in a real system, the user is able to change the system parameters without inter-

ruption of its operation. Thanks to that, it is possible to simulate the system immunity to the "interferences" due to its operation. The example response of ANC system to the compensated signal amplitude's change is presented in Fig. 7. The system is resistant to the compensated signal amplitude's change.

Figure 8 presents more complex ANC system structure. The system includes twelve objects:

1) `gen` – the compensated signal (the first harmonic component) generator, simulating noise source,
2) `gen2` – the compensated signal (the second harmonic component) generator,
3) `sum2` – the compensated signal component adder; the components are added before the signal is applied to the filter's input,
4) `del1` – the delay line that simulates the acoustic channel between the noise source and the compensating source; this line introduces the phase shift for the first harmonic component of the compensated signal,
5) `del2` – the delay line that simulates the acoustic channel between the noise source and the compensating source; this line introduces the phase shift for the second harmonic component of the compensated signal,
6) `sum1` – the adder for adding the compensated signal components in location of the secondary source,
7) `sum3` – the adder that corresponds to the location of the compensating source,
8) `fir` – the finite impulse response filter,
9) `del3` – the delay line that simulates the secondary signal path,
10) `xFil` – the delay line that compensates the effect of the secondary signal path,
11) `amp` – the amplifier that controls the compensating source,
12) `lms` – the LMS algorithm used for the digital filter coefficient adaptation.

Figure 8 structure's mapping in the simulation program is presented as the source code in Table 2.

Table 2. Mapping of ANC system structure presented in Fig. 8.

```
sum1.inp(1, gen1.out());
sum1.inp(2, gen2.out());
del1.inp(gen1.out());
del2.inp(gen2.out());
sum2.inp(1, del1.out());
sum2.inp(2, del2.out());
fir.inp(sum1.out());
sum3.inp(1, sum2.out());
amp.inp(fir.out());
del3.inp(amp.out());
sum3.inp(2, del3.out());
xfil.inp(sum1.out());
xbuf.inp(xfil.out());
xbuf.out(); // as this line is not connected
if(adapt)
lms.process(sum3.out());
```

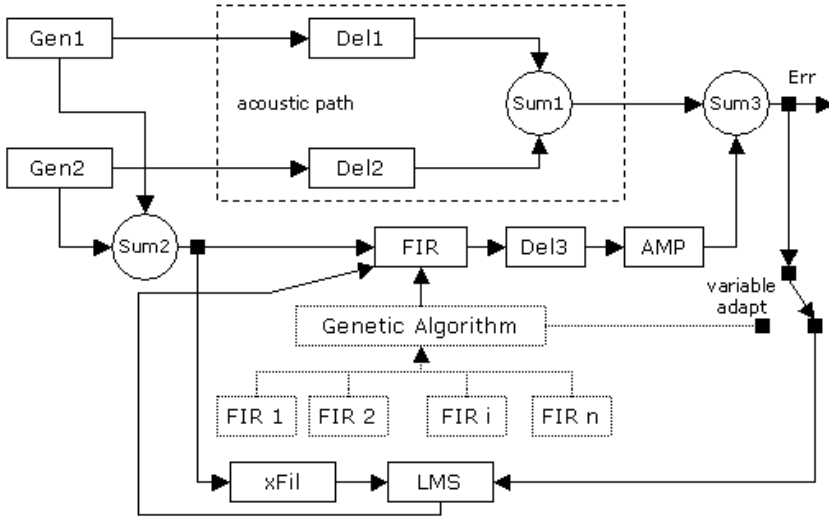

Fig. 8. ANC system structure.

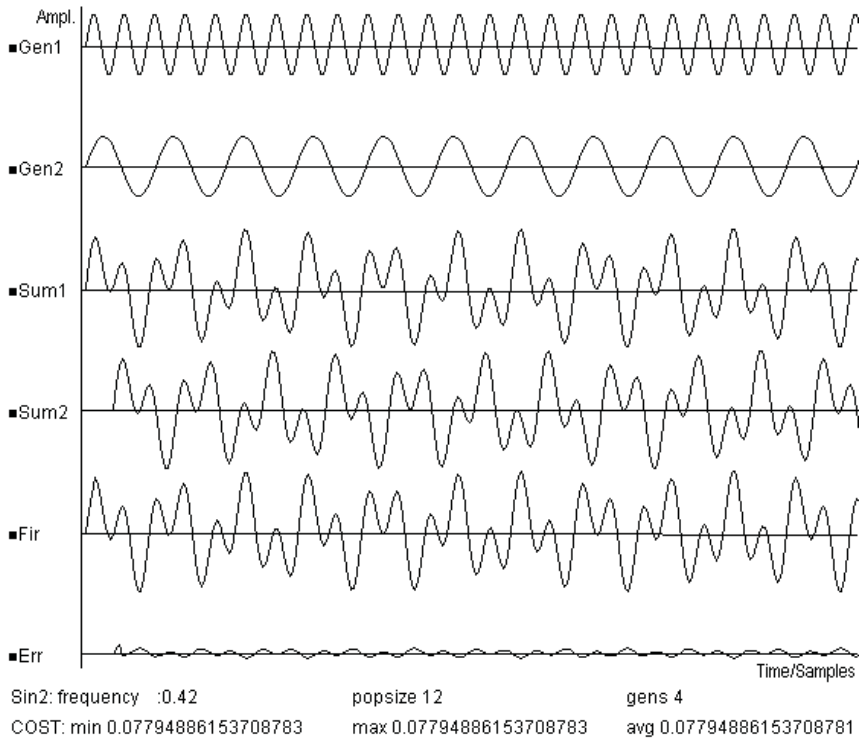

Fig. 9. The ANC system building block output signals (the system as of Fig. 8) (population size – 12, filter coefficients – 4, crossing probability – 0.5, mutation probability – 0.01).

Figure 9 presents example signal waves in the most important system points (Fig. 8). The calculations were performed for a small set including 12 digital FIR filters with 4 coefficients. The genetic algorithm operations were performed based on the mentioned heuristic crossing method.

The filter coefficient range was initially determined based on LMS algorithm, and then intentionally narrowed before the genetic algorithm operation. The waves in Fig. 9 present the error signal wave (Err) after 10 steps of the genetic algorithm operation. The crossing method enables compensating a two-tone signal, despite initial limitation of the base set filter's parameter values.

The presented programs show only a small part of the developed software capabilities. The user may perform the simulations by changing the parameter values and create own programs, basing on the developed classes set. As presented on the examples above, even complex ANC structure may be easily mapped and there is no formal limitation towards the model complexity.

5. Conclusions

The simulation calculations are a very important element of the genetic algorithm usage in ANC systems. In order to facilitate such research, the software for easy mapping of the structure of ANC systems and performing simulations related to the genetic algorithm application when searching for the optimal form of the ANC system's controller has been developed. The software includes broad set of functions related to the crossing operation. The heuristic crossing method, adapted especially for ANC systems, may be distinguished.

The simulation programs along with educational materials related to the genetic algorithms are presented on www.anc.pl. These programs may be used as practical tools enabling the genetic algorithm researching in extent of ANC systems. The class set, based on which the simulation programs were developed, enables easy development of simulation programs for examination of ANC system of any complexity level, so it may be very useful as a preliminary phase before designing ANC systems using the genetic algorithm. The object-oriented programming enables further enhancement of the software both by adding new classes to existing ones, and creating of child classes that inherit the properties of existing classes.

Acknowledgment

This publication has been prepared within the framework of disseminating the results of tasks of the National Programme "Adaptation of Working Conditions in Poland to European Union Standards", partly supported in 2002–2004 by the State Committee for Scientific Research of Poland – within the scope of research and development – and by the Ministry of Economy, Labour and Social Policy – within the scope of state services. The Central Institute for Labour Protection – National Research Institute has been the Programme's main coordinator.

References

[1] ARABAS J., *Wykłady z algorytmów ewolucyjnych*, Wydawnictwa Naukowo-Techniczne, Warszawa 2001.

[2] GOLDBERG D.E., *Algorytmy genetyczne i ich zastosowania*, Wydawnictwa Naukowo-Techniczne, Warszawa 1998.

[3] HANSEN C.H., *Active control of noise and vibration*, E and FN SPON, 1997.

[4] HUAMIN Y., HAICHAO Z., YIN S., *Active noise control in a practical air duct using genetic algorithm*, Active, 255–259 (2002).

[5] HAUPT R.L., HAUPT S.E., *Practical genetic algorithms*, John Wiley & Dons, Inc., 1998.

[6] MAKAREWICZ G., *Wybrane cyfrowe systemy aktywnej redukcji hałasu*, CIOP-PIB, Warszawa 2002.

[7] MAKAREWICZ G., MORZYŃSKI L., ZAWIESKA W., GÓRSKI P., WISZNIEWSKI M., *Badania możliwości wykorzystania algorytmów genetycznych w cyfrowych systemach aktywnej redukcji hałasu niskoczęstotliwościowego o charakterze stacjonarnym*, Sprawozdanie z realizacji zadania badawczego II-507, 2003.

[8] MAKAREWICZ G., ZAWIESKA W.M., *Zastosowanie algorytmów genetycznych do aktywnej redukcji hałasu*, Bezpieczeństwo Pracy, **4–6**, 1 (2003).

[9] MICHALEWICZ Z., *Algorytmy genetyczne + struktury danych = programy ewolucyjne*, Wydawnictwa Naukowo-Techniczne, Warszawa 1999.

[10] MINGUEZ A., RECUERO M., *Active noise control with simplified multichannel genetic algorithm*, International Journal of Acoustic and Vibration, **5**, 1 (2000).

[11] WERNER J.C., SOLETO J., LIMA R.G., FOGARTY T.C., *Active noise control in ducts using genetic algorithms*, Active, 243–254 (2002).

[12] WRIGHT A., *Genetic algorithm for real parameter optimization*, Morgan Kaufmann, 1991.