Exercise 3

Investigation of Adaptive Filters

László Sujbert, Tibor Balogh
BME Department of Measurement and Information Systems
English translation: András Palkó

1 Aim of the Measurement Exercise

The purpose of this measurement exercise is to deepen the knowledge acquired during previous specialized studies about adaptive filters belonging to the topic of digital signal processing, and to learn about their practical aspects. In this measurement exercise the deepening of existing theoretical knowledge is followed by the examination of their practical applications.

According to today's typical trends, the practical application of adaptive signal processing is spreading more and more, so this topic expands the toolbox of the electrical engineer not only on a theoretical level, but also on a practical level. Nowadays, the computing power of the signal processors offered in the product range of manufacturers is increasing, and at the same time, their prices are getting lower, so their prevalence is also showing an increasing trend, and they are also gaining ground in areas where they were not used until now due to their high prices or insufficient complexity. Today's modern signal processing processors therefore offer the opportunity to solve practical problems that were previously unattainable and unimaginable – among other things – due to the need for complex real-time signal processing.

Today, there is a need and opportunity to incorporate adaptive signal processing in more and more applications. The largest group of these is closely related to acoustics or the transmission of audio signals. The transfer function of the acoustic space typically varies between two physical locations, such as speakers and microphones, or natural noise sources and the human ear. The transfer function of the transmission channel can also change significantly due to changes in the parameters of equipment that is often physically far from each other.

However, according to the considerations of many algorithms used in practical tasks, it is necessary to know some transfer functions with sufficient accuracy. Based on the previous aspects, this is not a straightforward task, given the stochastically variable nature of the transfer functions. It must therefore be possible in some way to produce a model of the transfer function of an unknown and time-varying system with satisfactory accuracy, as well as for the model to follow the changes of the physical system in real time with sufficient speed.

The task is therefore model fitting, which includes the parametric or structural definition of the model, or the creation of a model with similar behavior to the system but with a different structure. During the exercise, these goals are served on the one hand by MATLAB simulations, and on the other hand by measurements performed with the help of a circuit and a signal processing card.

2 Theoretical Grounds of the Measurement Exercise

In order to complete the measurement exercise, a comprehensive knowledge of the theoretical background of adaptive signal processing is definitely necessary. In some of the tasks, the implementation of signal processing algorithms in MATLAB must be carried out, while in the other tasks, measurements and parameter tuning tasks must be performed on physically realized structures.

2.1 Theory of Model Fitting

The task of model fitting – summarized in one sentence – is to determine the structure of a model corresponding to an unknown system, and, where applicable, to determine the parameters of the previously assumed known structure (*parameter estimation*). In practice, we mostly encounter the task of parameter estimation, during which we look for a model with parameters that best fit the unknown system, with the structure given in advance.

During the parameter estimation, we want to minimize the deviation of the outputs of the real system and the model fitted to it based on a given error criterion. The scope of tasks of model fitting can also be divided into two groups. The purpose of *identification* is to determine the parameters of the system assumed to be constant over time, while the task of *adaptation* is to follow system parameters that change over time. Therefore, during

the identification, it is theoretically possible to carry out high-precision measurements, because the system can be observed for the required long time, and a large amount of data can be collected from it. Consequently, identification is typically a time-consuming process. During adaptation, high accuracy and the exact fit of the model to the real system are less important. It is much more crucial that we can follow the changes of the physical system by changing the parameters of the model, and that the model can vary together with the system in real time. Of course, the structure of the physical system is not always known, and its parameters cannot always be measured: only the difference between the outputs of the real system and the model can help us to tune the model parameters. Thus, in the vast majority of cases, we have to give up the high-precision model and instead use a simpler structure that is easier to handle in practice, which is still acceptable for modeling the real system.

A special case of model fitting is the task of regression, the purpose of which is to determine the deterministic relationship between certain variables. In practice, these variables are mostly the inputs and outputs of certain systems. The real systems to which we want to fit a model, however, do not implement a purely deterministic relationship between their input and output, because the output usually contains some noise, so it also has a stochastic component in addition to the deterministic one. On the other hand, the model to be fitted only realizes a deterministic relationship, the design of which typically means the setting of certain free parameters based on a principle chosen from some point of view using the series of input-output data pairs available after the measurements.

To tune the parameters, we define a so-called *cost function* depending on the outputs of the real system and our model, then try to minimize it by setting the model's parameters. Our goal is therefore to achieve that the output of the model approximates the output of the real system as closely as possible, and the exact description of the "as closely as possible" criterion is the expression of the cost function or error criterion function:

$$\varepsilon = \mathrm{E}\{(\mathbf{y} - \hat{\mathbf{y}})^T(\mathbf{y} - \hat{\mathbf{y}})\}. \tag{1}$$

An advantageous choice of the cost function is if its value depends on the square of the difference of the outputs, i.e. the error of the model, because finding the minimum of such a cost function is mathematically simple, and in addition, we physically minimize the power of the error signal. A filter whose coefficients are tuned in order to minimize the value of a cost function defined in this way is called a *Wiener filter*.

However, in order to carry out the parameter estimation, it is not enough to know the instantaneous values of the measured value pairs, it is necessary to know their statistical characteristics, or at least a certain degree of their moments. When applying linear regression, for example, it is sufficient if the first and second order moments are available.

In particular, if the model to be fitted is a linear function of the parameters and a quadratic cost function is used, the error function is a surface stretched over the plane of the parameters, and the optimal value of the parameters is immediately obtained after searching for its minimum. However, this requires full knowledge of the error surface.

The relationship between the input, output and parameters of the model to be fitted is as follows:

$$\hat{\mathbf{y}} = \hat{\mathbf{g}}(\mathbf{u}) = \hat{\mathbf{g}}(\mathbf{W}, \mathbf{u}), \tag{2}$$

where $\mathbf{u}$ is the input signal vector, $\mathbf{W}$ is the matrix of adjustable parameters, and $\hat{\mathbf{y}}$ is the output of the model. The input and output, as well as the output of the model, are all certain realizations of at least weakly stationary stochastic processes. The notation expresses that the model can be adapted via the parameter matrix $\mathbf{W}$.

It is advisable to fix the dynamic and potentially nonlinear part of the model ($f(\mathbf{u}) = \mathbf{x}$) and separate it from the linear, adaptable part ($\mathbf{W}$):

$$\hat{\mathbf{y}} = \hat{\mathbf{g}}(\mathbf{W}, \mathbf{u}) = \mathbf{W}^T f(\mathbf{u}) = \mathbf{W}^T \mathbf{x}, \tag{3}$$

where $\mathbf{x} = f(\mathbf{u})$ is the fixed part of the model, which generates the regression vector $\mathbf{x}$ from the input signal vector $\mathbf{u}$, and then forms the input of the adaptive part. In the case of a single output, $\hat{\mathbf{y}}$ is the scalar product

of two vectors, but in the case of multiple outputs, it appears as a matrix-vector product, as can be seen from the formula. The form of the expressions for one input and one output:

$$\hat{y} = \hat{g}(\mathbf{w}, u) = \mathbf{w}^T f(u) = \mathbf{w}^T \mathbf{x}. \tag{4}$$

Moving from stochastic processes to time functions (realizations of the processes):

$$\hat{y}(n) = \mathbf{w}^T(n)\mathbf{x}(n) = \sum_{i=1}^{M-1} w_i(n) x_i(n), \tag{5}$$

where M is the length of the vectors $\mathbf{w}$ and $\mathbf{x}$, and the matrix-vector product is simplified to a scalar product of vectors and can be calculated based on (5).

As a result of the separation, the adaptive part is linear in its parameters, and the transient caused by the change of parameters runs out of the system in a finite time.

Since we have already written the expression of the model's output using the regression vector $\mathbf{x}$ and the parameter vector $\mathbf{w}$ (4 and 5), let us write the criterion function used for model fitting (1) based on the mean squared error:

$$\varepsilon(n) = \mathrm{E}\{(y(n) - \hat{y}(n))^2\} = \mathrm{E}\{(y(n) - \mathbf{w}^T\mathbf{x}(n))^2\} = \tag{6}$$

$$= \mathrm{E}\{y^2(n)\} - 2\underbrace{\mathrm{E}\{y(n)\mathbf{x}^T(n)\}}_{\mathbf{p}^T}\mathbf{w} + \mathbf{w}^T\underbrace{\mathrm{E}\{\mathbf{x}(n)\mathbf{x}^T(n)\}}_{\mathbf{R}}\mathbf{w} = \tag{7}$$

$$= \mathrm{E}\{y^2(n)\} - 2\mathbf{p}^T\mathbf{w} + \mathbf{w}^T\mathbf{R}\mathbf{w}, \tag{8}$$

where $\mathbf{p}^T$ is the cross-correlation vector between the output and the regression vector, and $\mathbf{R}$ is the autocorrelation matrix of the regression vector. The expression of the criterion function according to (8) describes a paraboloid located in $M+1$-dimensional space. The minimum of the criterion function by differentiating according to the M-dimensional vector $\mathbf{w}$ results in the following:

$$\frac{\partial \varepsilon}{\partial \mathbf{w}} = -2\mathbf{p} + 2\mathbf{R}\mathbf{w} = 2(\mathbf{R}\mathbf{w} - \mathbf{p}) = \mathbf{0}, \tag{9}$$

whose solution is the Wiener-Hopf equation, which gives the location of the optimum in the space of $\mathbf{w}$ filter coefficients:

$$\mathbf{w}_{opt} = \mathbf{R}^{-1}\mathbf{p}. \tag{10}$$

Determining the optimal set of parameters based on these is an extreme value problem. The difficulty of the result is that in order to calculate the $\mathbf{w}_{opt}$ set of coefficients, the statistical properties of the input and output signals of the system to be modeled must be known *a priori* to determine the correlation vectors and matrices.

In the absence of knowledge of the correlation vectors, or in the case of partial knowledge of them, the parameter setting can only be done iteratively. Then, instead of statistical parameters, we try to approach the optimal set of parameters by using the instantaneous values of the signals. In the individual steps of the iteration, the gradient of the error surface at the given point must be determined, and the parameter set for the next step must be designed based on this knowledge.

A possible approach is the steepest descend method, in which we move in the steepest direction, along the negative gradient, and the length of the step is dictated by the so-called learning rate or adaptation constant. By using this, we can avoid performing complicated calculations, as well as eliminate the lack of prior knowledge of statistical data.

2.2 Adaptive Signal Processing Algorithms

2.2.1 The LMS Algorithm

The knowledge requirement of statistical characteristics and complicated calculations can be eliminated, if instead of the average error only the instantaneous error is taken into account in the cost function, and the

steepest descent method is modified accordingly. This is how the LMS[1] algorithm works when adapting the model parameters. The autocorrelation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}^T$ are therefore modified as follows:

$$\mathbf{R} = \mathrm{E}\{\mathbf{x}(n)\mathbf{x}^T(n)\} \text{ changes to } \hat{\mathbf{R}} = \mathbf{x}(n)\mathbf{x}^T(n), \tag{11}$$

$$\mathbf{p}^T = \mathrm{E}\{y(n)\mathbf{x}^T(n)\} \text{ changes to } \hat{\mathbf{p}}^T = y(n)\mathbf{x}^T(n). \tag{12}$$

However, due to the estimation based on the instantaneous error, the modification of the parameters will be inaccurate, not necessarily aligning with the direction of the negative gradient. However, in a longer time interval, the inaccuracy of individual modifications is averaged out. On the other hand, using this method, the model can better track the parameter changes of the estimated system. The peculiarity of the LMS algorithm is that the instantaneous error around the minimum is small, and the gradient of the error surface is around zero. Thus, the set of parameters never stabilizes at the minimum value, because it does not even reach it, but varies in a small environment around it in each step. In the LMS algorithm, based on these, it is advisable to choose the size of the steps in the parameter space, the *learning rate*, to be small. However, it deteriorates the settling properties of the algorithm. The optimal value of the learning rate μ can be determined by knowing the autocorrelation matrix $\mathbf{R}$.

The block diagram of the simplest structure using the LMS algorithm is shown in Fig. 1.

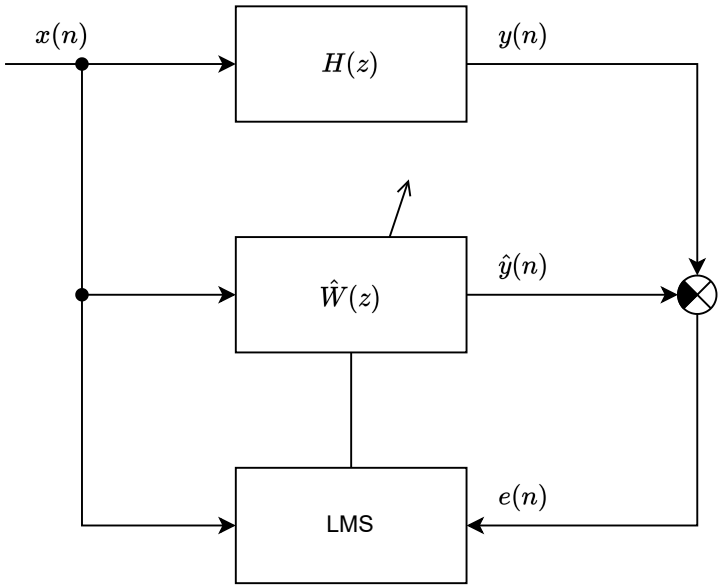

Figure 1: Block diagram of the LMS algorithm

The recursive equations to be calculated in the LMS algorithm for the error and the coefficients are:

$$e(n) = y(n) - \hat{\mathbf{w}}^T(n)\mathbf{x}(n), \tag{13}$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + 2\mu e(n)\mathbf{x}(n). \tag{14}$$

The model used by the LMS algorithm, whose parameters are tuned, is a finite impulse response (FIR) digital filter whose impulse response is the vector $\mathbf{w}$. The set of filter coefficients available in the nth step is given by the vector $\hat{\mathbf{w}}(n)$. The LMS algorithm is therefore able to produce filter coefficients approximating the coefficient set $\mathbf{w}_{opt}$ without knowing the autocorrelation matrix $\mathbf{R}$ and the cross-correlation vector $\mathbf{p}^T$.

[1] Least Mean Squares

2.2.2 The NLMS Algorithm

If we want to improve the stability and settling properties of the LMS algorithm, it is obvious that the learning rate should be tuned in opposite directions according to the two aspects. A possible solution to this problem is offered by the NLMS[2] algorithm. NLMS normalizes the learning rate based on the input signal, so it has better stability properties and generally faster settling under given conditions. Without normalization, the settling speed of the LMS algorithm depends squarely on the amplitude of the reference signal. Normalization eliminates this, i.e. the convergence will be independent of the amplitude of the excitation of the system (the reference).

The recursive equations of the NLMS algorithm regarding the error and the coefficients are as follows:

$$e(n) = y(n) - \hat{\mathbf{w}}^T(n)\mathbf{x}(n), \tag{15}$$

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \frac{\tilde{\mu}}{a + \mathbf{x}^T\mathbf{x}} e(n)\mathbf{x}(n). \tag{16}$$

The function of the regularization constant a is to prevent the denominator of the fraction from becoming close to zero even with small inputs, causing too large steps.

2.2.3 The FxLMS Algorithm (Supplementary Material)

In some practical applications, due to the physical structure of the system, another system with non-uniform transfer function is also connected to the output of the filter with a set of parameters adapted using the LMS algorithm, and its output is used when the difference signal is formed. In these cases, it can no longer be assumed that the output of the adaptive filter can be immediately subtracted from the output of the identified system, and that the difference signal produced in this way can be used directly in the adaptive algorithm to tune the adaptive filter.

In other words, looking at the block diagram, another system is inserted between the output of the adaptive filter and the subtraction block. The biggest problem is that in addition to frequency domain filtering, it realizes a phase shift and delay between the output of the adaptive filter and the subtraction. So it can easily cause instability if due to phase shift, we modify the filter coefficients not in the direction of the negative gradient. The filter inserted in this way in the path of the estimated output signal is called "secondary path" in the literature, mainly dealing with active noise reduction (the "primary path" means the system to be modeled).

Therefore, if the system to be identified reacts to a specific input signal with some delay, the LMS algorithm tries to tune the filter to match the system based on the observation of the input and output, but does not take into account that an inserted transfer function modifies the filter output. The effect of the so-called secondary transfer function is eliminated if the adaptive filter models the inverse of the secondary transfer function in addition to the identified system. In extreme cases, if the delay of the inserted system is greater than the delay of the original system, this task cannot be solved. If the secondary transmission realizes a significant phase shift at certain frequencies, it leads to instability of the control loop.

There are several theoretical solutions to eliminate the problem. It is a theoretical idea that the effect of the secondary path can be eliminated by placing its inverse in series. This is not necessarily feasible in practice, because the inverse of the transfer function may not exist. The practical solution is offered by the structure implemented by the FxLMS[3] algorithm. According to this approach, the estimate of the secondary path must be placed at the input of the LMS algorithm in such a way that it filters the reference signal, but does not affect the input of the filter to be adapted. This structure can be seen in Fig. 2. Based on the linearity of the system, the transfer function of the secondary path can also be imagined in front of the adaptive filter, so it becomes obvious why it is necessary to filter the reference signal with this transfer function. However, to achieve this, it is necessary to know the secondary path with sufficient accuracy, for which a possible method is the application of the LMS algorithm.

[2] Normalized LMS
[3] Filtered-x LMS

In the FxLMS algorithm, the error signal is as follows:

$$e(n) = y(n) - \hat{y}_s(n), \tag{17}$$

where $\hat{y}_s(n)$ is the filtered version of $\hat{y}(n)$ signal by the $S(z)$ secondary path. Let the block with transfer function $C(z)$ be the estimator of the transfer function $S(z)$, and let $\mathbf{c}$ be its finite impulse response (the vector contains the FIR filter coefficients). Then $e(n)$ can be written as follows:

$$e(n) = y(n) - \hat{\mathbf{w}}^T(n)\mathbf{x}_c(n), \text{ ahol} \tag{18}$$

$$\mathbf{x}_c(n) = \begin{bmatrix} \sum_{i=0}^{I-1} c_i x(n-i) \\ \sum_{i=0}^{I-1} c_i x(n-i-1) \\ \dots \\ \sum_{i=0}^{I-1} c_i x(n-i-(M-1)) \end{bmatrix}. \tag{19}$$

The vector $\mathbf{c} = [c_1, c_2, \dots, c_I]$ is the (finite) impulse response of the secondary path estimator, and I is the length of $\mathbf{c}$.

The recursive equation that can be used to adapt the filter coefficients:

$$\hat{\mathbf{w}}(n+1) = \hat{\mathbf{w}}(n) + \mu e(n)\mathbf{x}_c(n). \tag{20}$$

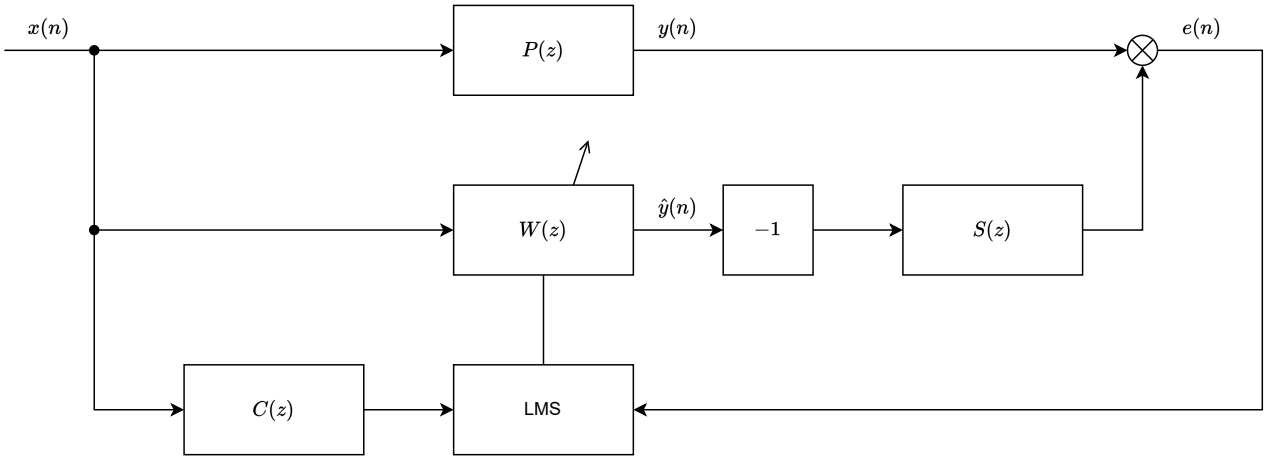

Figure 2: Block diagram of the FxLMS algorithm

3 Structures Using Methods of Adaptive Signal Processing

3.1 Adaptive Line Enhancer

One of the main areas of adaptive digital signal processing aims to suppress various noises. One of the possible frequent problems is, for example, to filter out additive noise burdening an information-carrying signal. The Adaptive Line Enhancer (ALE), is a structure that implements such a task. Its application may be necessary in cases where the bandwidth and other parameters of the narrow-band signal of interest signal change over time, while we have little prior (*a priori*) knowledge about them. Such areas of application can be, for example, sonar, certain medical applications and various speech-related processing.

The ALE is a special noise suppression structure that allows for adequate damping and suppression of the noise burdening its input, while the input component carrying the useful signal is ideally amplified by unity, or at least with a small attenuation. Due to adaptive signal processing, it is not necessary for the signal to have stationary properties.

Regarding the structure of the ALE, it is built by connecting a delay and a linear predictor, as shown in Fig. 3. The input signal $y(n)$ of the ALE consists of the useful signal $u(n)$ and the additive noise $v(n)$. By subtracting the output signal $\hat{y}(n)$ of the predictor from the input signal $y(n)$, we obtain the estimation error $e(n)$, which is used for adaptive tuning of the predictor. In our case, the linear predictor is implemented by an FIR filter, the coefficients of which are tuned by the LMS algorithm.

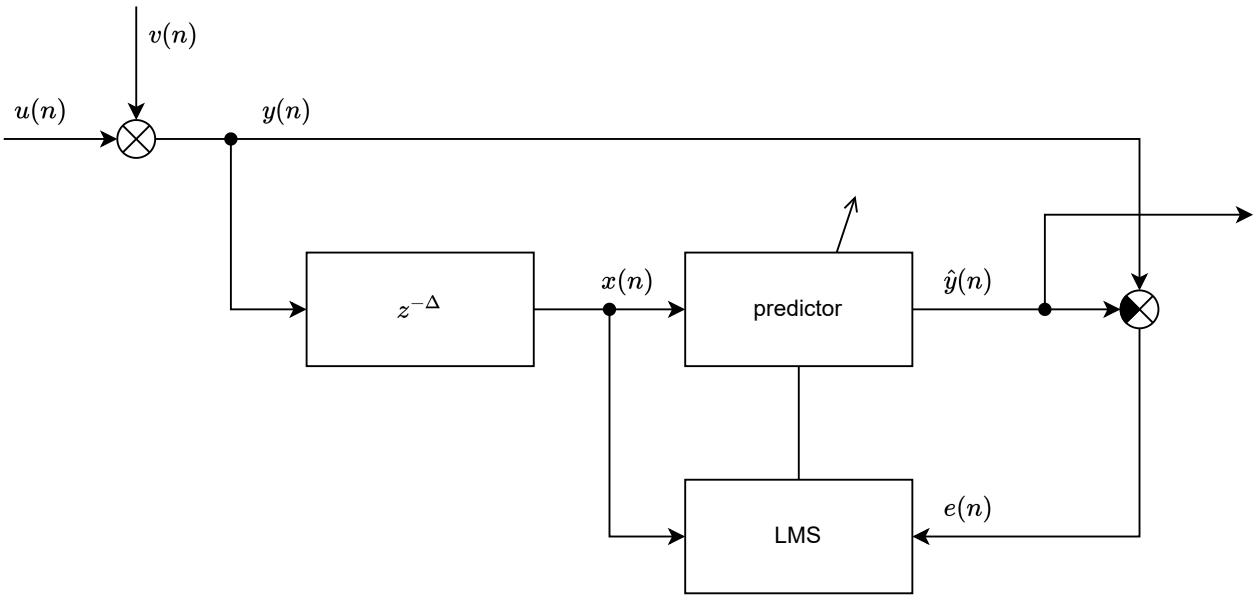

Figure 3: Block diagram of the ALE structure

The input signal $x(n)$ of the predictor is the delayed version of the ALE input by Δ number of samples:

$$x(n) = y(n - \Delta). \tag{21}$$

The amount of delay Δ of the structure should be chosen according to the noise component $v(n)$. If the noise component is broadband, the delay should be chosen so large that $v(n)$ and its delayed version $v(n-\Delta) \ldots v(n-\Delta-M)$ are uncorrelated (M denotes the length of the FIR filter). In this way, when the estimation error is formed, the noise component of the input and the noise component of the adaptive filter output originated from the incoming noise will be uncorrelated. As a result, during adaptation, it does not happen that the adaptive algorithm adjusts the filter coefficients in such a way that the filter also lets noise through in order to minimize the estimation error. In this case the output signal of the structure is the signal $\hat{y}(n)$. However, if the delay Δ is chosen too large, the samples of the useful (nonstationary) signal will also be uncorrelated, and the structure will not work.

If the noise component is narrowband, the noise and its samples available with a very long delay are also correlated (in case of periodic noise, the delay can be arbitrarily large). In this case, the delay should be chosen so large that $u(n)$ and its delayed version $u(n-\Delta) \ldots u(n-\Delta-M)$ are uncorrelated (M denotes the length of the FIR filter). In this way, when forming the estimation error, the input signal and the adaptive filter output component from the useful signal will be uncorrelated. In this case, the output signal of the structure is the signal $e(n)$.

So if the Δ value of the so-called decorrelation delay is adequate, it becomes possible to cancel the noise, but the choice of the delay and the output depends on the characteristics of the noise to be suppressed.

3.2 Adaptive Echo Reduction

The echo problem first appeared most widely in the PSTN[4] telephone networks. Due to the large distance between the hybrids located in the switching centers and the imperfect matching, reflection occurs, which can cause an audible echo due to the high delay, which above a certain limit disturbs the speakers, or can even significantly impair speech intelligibility.

The model of the adaptive echo reduction structure is shown in Fig. 4. The speaker-microphone pair shown on the left side of the figure belongs to one person, while the pair shown on the right is the endpoint belongs to the other person. The dashed lines illustrate the long transmission path, but from the point of view of the echo, the delay caused by them is included in the transfer function $H(z)$, which represents the transfer function of the hybrid located on the listener's side.

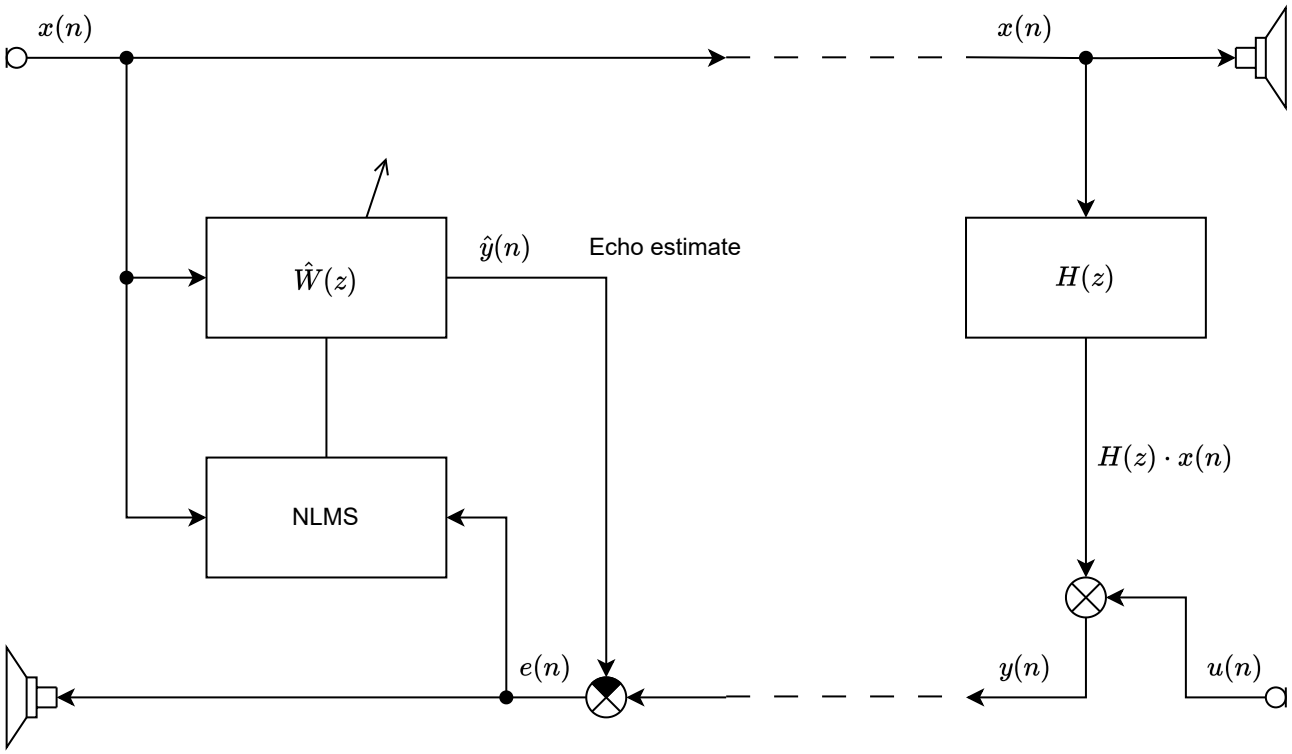

Figure 4: Block diagram of the adaptive echo cancellation structure

The speaker hears its own echo if $H(z) \neq 0$. Consider the case when the listener is silent (i.e. $u(n) = 0$). In this case, an echo $y(n) = H(z) \cdot x(n)$ is sent back to the speaker from the listener's side. In Fig. 5, we can see the right side of Fig. 4. redrawn for this case. It can already be seen that the adaptive echo reduction is nothing else than the identification task of the hybrid located on the listener's side.

If the echo can be estimated in a suitable way, the estimated echo signal $\hat{y}(n)$ can be subtracted from the signal $y(n)$ returning to the speaker's side. The difference between the two signals is the estimation error signal: $e(n)$. The estimation is done with the NLMS algorithm described in Section 2.2.2, which uses the error signal $e(n)$ and the speaker's signal $x(n)$. It is important that all the signals required by the algorithm are available in one place, in this case on the left side of the figure.

[4]Public Switched Telephone Network

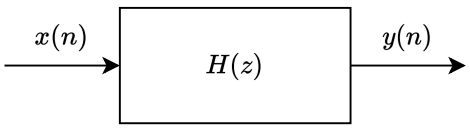

Figure 5: When $u(n) = 0$, the listener-side hybrid is identified

3.3 Active Noise Control (Supplementary Material)

The task of active noise control is to suppress the noise coming from the environment or some noise source with counter-phase noise. Active suppression of acoustic noise is such an application that creates an independent field of expertise. In this case, it is necessary to place loudspeakers in suitable places, and with the counter-phase noise emitted by them, quiet zones can be created in certain surroundings of the error microphones. Acoustic systems can be considered linear with a good approximation in a large dynamic range, so the principle of superposition can be effectively applied. In addition to acoustic applications, active noise reduction can also be effectively used to dampen mechanical vibrations, and the appropriate sensors and actuators are also available in this case (for example, acceleration sensors and shakers).

Fig. 6 shows the block diagram of an active noise reduction system, which suppresses acoustic noise. On the left side of the figure is the noise source, the sound of which spreads through the air, and which is directly connected to the DSP by the so-called reference microphone. The so-called error microphone also records the noise. The transfer function from the noise source to the error microphone is the *primary path*, while the transfer

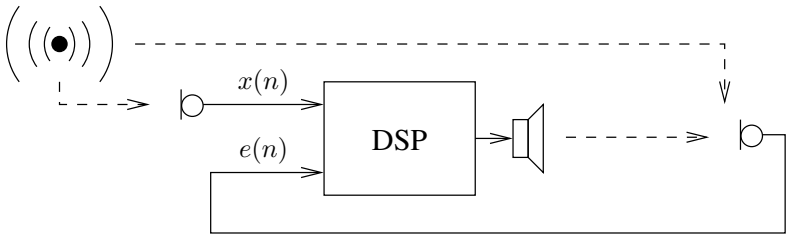

Figure 6: Active noise control system

function from the loudspeaker to the error microphone is the *secondary path*, following the terminology used in the literature.

The figure representing the physical layout can be compared with the theoretical block diagram, which is the FxLMS algorithm (Fig. 2). The transfer function between the reference signal $x(n)$ and the noise to be suppressed $y(n)$ is $P(z)$. (Here, the transmission between the noise source and the reference signal is considered constant.) The transfer function $S(z)$ means the transfer function between the output signal $\hat{y}(n)$ of the adaptive filter and the error signal $e(n)$. The phase reversal is necessary because in the physical system, superposition only performs addition. $S(z)$ represents not only the acoustic transmission, but also all devices in the signal path (amplifiers, AD and DA converters, etc.). The signals are not summed in the DSP and are not directly accessible. That is why $S(z)$ cannot be equal to unity, so the FxLMS algorithm must be used for adaptation.

The secondary path can be identified with the LMS algorithm. The input of the system to be identified is the signal to be sent to the speaker, and the output is the signal of the error microphone. In the case of suitable excitation (e.g. white noise), after settling, the coefficients representing the $\hat{W}(z)$ transfer function of the adaptive filter must be copied to the place of the coefficients of $C(z)$, thus $C(z)$ will truly be the estimator of the secondary path $S(z)$.

4 The Device Used in the Practical Measurement Tasks

During the exercise, measurements must be performed on an electronic hybrid pair. The simplified wiring diagram of the circuit is shown in Fig. 7. In the circuit, the signal given to the `IN` input is transmitted to the

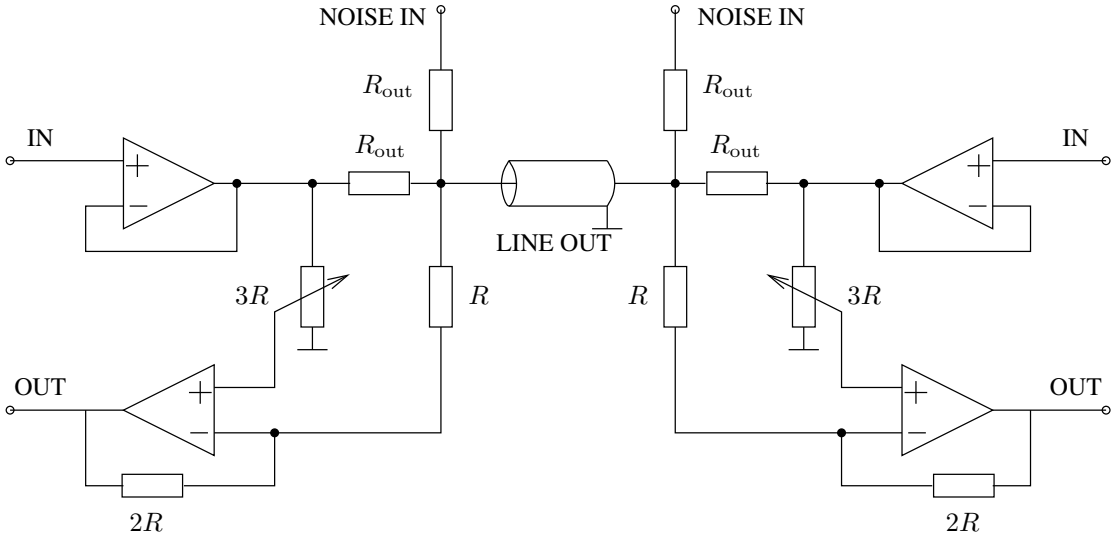

Figure 7: The electronic hybrid circuit to be measured

`LINE OUT` line output, and the signal coming from `LINE OUT` can be accessed via the `OUT` output. Noise can be added to the channel at the `NOISE IN` input. The two sides of the hybrid pair can be connected using a coaxial cable. In the circuit $R_{\text{out}} \ll R$. If the potentiometer is set in such a way that it divides the voltage of the operational amplifier driving the line by three, the circuit suppresses the signal given to the `IN` input on the same side, while the signal arriving on the line is transmitted with unit gain. (We leave its proof to the discretion of the Reader.) However, due to the tolerance of the circuit elements and changes in physical conditions, this is not exactly fulfilled, and in reality, on the receiving side, acoustic coupling may occur between the output and input, which is why adaptive echo reduction is necessary, which is done using a DSP card.

The noise applied to the `NOISE IN` input can be compensated. On the one hand, it is possible to operate an adaptive line enhancer on the `OUT` output, and on the other hand, active noise reduction can be implemented with the noise applied to the `IN` input. Then the error signal appears on the `OUT` output. In this case, the dynamics of the secondary path are mainly formed by the AD and DA converters.

5 The Signal Processing Board and the Development Environment

5.1 Digital Signal Processors

DSPs[5], i.e. digital signal processing processors, are widespread in the field of embedded systems. They can be found in mobile phones, audio and video, DVD players, digital cameras and camcorders, and many industrial applications where you need to work with signals or data streams. In contrast to general-purpose processors, DSPs are optimized for real-time signal processing applications both at the architectural level and in the design of their instruction set. They are designed in such a way that efficient signal processing can be done with the execution of as few instructions as possible.

[5] Digital Signal Processor

5.2 The ADSP-21364 EZ-KIT Lite Development Board

The ADSP-21364 EZ-KIT Lite development board is an efficient and fast tool for development with the ADSP-21364 signal processor. All necessary peripherals and environments of the DSP, which may be needed when performing the most common tasks, are accessible on the development card. Analog and digital audio inputs and outputs, LEDs, switches, and buttons are accessible, so software development can begin immediately without any hardware design steps. When designing the development card, they tried to make available as many use cases as possible, and to make the functionality provided by the processor as thoroughly understandable as possible. The card provides the developer with the following hardware elements:

- Analog Devices ADSP-21364 processor
- 512 kbit $\times$ 8 bit SRAM[6]
- 1 Mbit $\times$ 8 bit flash memory
- 2 Mbit flash memory accessed by SPI[7]
- Analog sound interface (AD1835A codec, 1 stereo input, 4 stereo output)
- Digital sound interface (1 input, 1 output)
- 11 LED (of which 1 "power", 1 "board reset", 1 "USB monitor", 8 general purpose)
- 5 button (1 "reset", 2 on DAI pin, 2 on FLAG pin)
- Extension interface (parallel port, FLAGs, DAI, SPI)
- JTAG emulator port
- USB port for PC connection.

The image of the board is shown in Fig. 8, and the side view of the connectors is shown in Fig. 9.

5.3 The CrossCore Embedded Studio Development Environment

The development environment is used for multiple laboratory exercises, so its description can be found in a separate document. During this measurement, there is no need to solve a real development task, you have to run pre-made programs or modify the contents of the memory in order to change the parameters.

6 Measurement Tasks

6.1 Simulations in MATLAB

6.1.1 Implementation of the LMS Algorithm

1. According to the equations (13) and (14) in the theoretical overview, write a MATLAB function for the implementation of the LMS algorithm performing the system identification task according to Fig. 1. The skeleton of the function that implements the algorithm should be as follows:

```
function     [e,w]=lms(mu,M,x,y);
             %input parameters:
             %mu:  step size (learning rate) 1 × 1
             %M: number of filter coefficients 1 × 1
```

[6] Synchronous Random Access Memory
[7] Serial Peripheral Interface

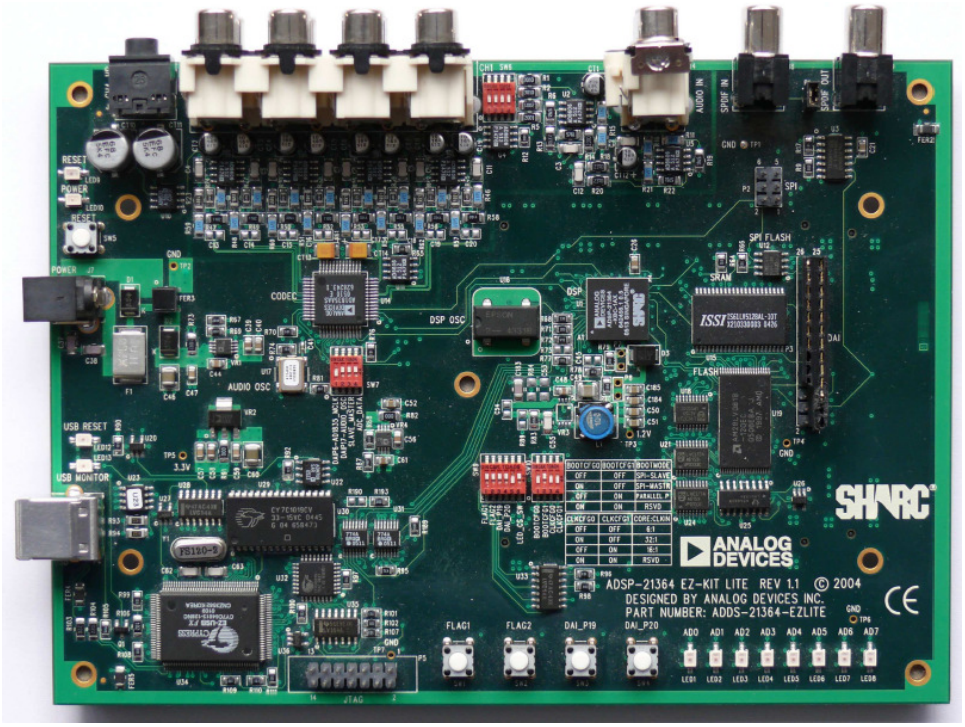

Figure 8: Image of the ADSP-21364 signal processing board

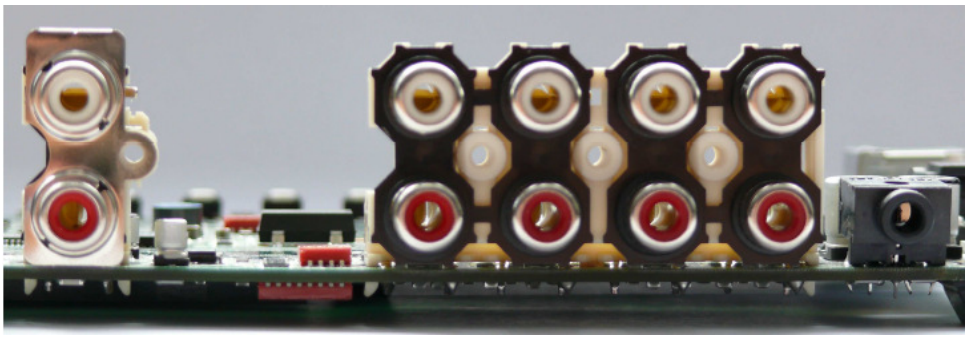

Figure 9: Image of ADSP-21364 signal processing board connectors

```
%x:  input signal of the filter N × 1
%y:  output signal of the filter N × 1

%output variables:
%e:  error signal N × 1
%w:  filter coefficients M × 1
```

2. Design an IIR filter according to the instructions of the teacher.
3. Identify the IIR filter using the LMS algorithm. Use white noise with Gaussian distribution as excitation. Compare the transfer function of the IIR filter and the FIR filter resulting from the identification.
4. Try different values of μ and M and see how they affect the convergence of the adaptive filter and the goodness of the approximation of the IIR filter.

6.1.2 Implementation of the NLMS Algorithm

1. According to the equations (15) and (16) in the theoretical overview, write a MATLAB function for the implementation of the NLMS algorithm performing the system identification task according to Fig. 1. The skeleton of the function that implements the algorithm should be as follows:

```
function     [e,w]=nlms(mu,a,M,x,y);
             %input parameters:
             %mu:  step size (learning rate) 1 × 1
             %M: number of filter coefficients 1 × 1
             %x:  input signal of the filter N × 1
             %y:  output signal of the filter N × 1
             %a:  regularization constant 1 × 1

             %output variables:
             %e:  error signal N × 1
             %w:  filter coefficients M × 1
```

2. Try different excitation signal powers and μ values and see how they affect the convergence of the adaptive filter. Identify the IIR filter designed in the previous task and compare the results with those obtained there.

6.1.3 Implementation of the FxLMS Algorithm (Optional)

1. According to the equations (17), (18), (19) and (20) in the theoretical overview, write a MATLAB function for FxLMS algorithm according to Fig. 2. The transfer function $S(z)$ should be a simple FIR filter designed for this task. (Since the latter filter is also part of the adaptation loop, $\hat{y}_s(n)$ cannot be generated blockwise, i.e. the previous `lms` function cannot be used directly.) The skeleton of the function implementing the algorithm should be as follows:

```
function     [e,w]=fxlms(mu,M,s,c,x,y);
             %input parameters:
             %mu:  step size (learning rate) 1 × 1
             %M: number of filter coefficients 1 × 1
             %x:  input signal of the filter N × 1
             %y:  output signal of the filter N × 1
             %s:  coefficients of the secondary path (FIR filter) S × 1
             %c:  coefficients of the secondary path estimate (FIR filter) C × 1

             %output variables:
             %e:  error signal N × 1
             %w:  filter coefficients M × 1
```

2. Identify the transfer function $S(z)$ with the previously written LMS algorithm, use the identified coefficients in the filter $C(z)$.

3. Test the FxLMS algorithm with different reference signals (sine, triangle, etc.). Let the transfer function $P(z)$ be the IIR filter used previously.

4. Create a simulation that demonstrates that if $S(z)$ is insufficiently identified, the algorithm is not convergent, even if μ is arbitrarily small!

6.2 Experiments with the Signal Processor

To solve the tasks, the ADSP-21364 EZ-KIT Lite development board described above and the electronic hybrid circuit must be used. It is advisable to listen to the output signals on a loudspeaker or headphones, and to analyze them visually on an oscilloscope.

6.2.1 Adaptive Line Enhancement

To solve the task, you need a source that can be used in the sense of "useful" or "noise". The useful signal should be an audio signal (radio, MP3 player, etc. output signal), the noise can be generated using a function and noise generator.

The `SHARC_Lab_ALE` project must be downloaded to the DSP board. Using the markings shown in Fig. 3, during the measurement, the input signal $y(n)$ of the ALE structure must be connected to the left channel (marked in red) of the input of the signal processing card. The signals $y(n)$, $\hat{y}(n)$, and $e(n)$ can be listened to and processed on the outputs of the card:

- y: Left and right channels of output 1
- $\hat{y}$: Left and right channels of output 2
- e: Left and right channels of output 3
- x: Left and right channels of output 4

1. Use the hybrid to create the sum of the useful signal and the noise. There are multiple options for this, make sure that neither the circuit nor the source gets loaded significantly.
2. Operate the structure using periodic noise. Try to find the optimal amount of the delay Δ.
3. Make the structure work using random noise. Try to find the optimal amount of the delay Δ.
4. Try different values of μ and investigate the settling and stability properties of the system. How does changing μ affect the useful signal?

6.2.2 Adaptive Echo Reduction

Two signal sources are needed to solve the task. One source must be the audio signal used in the previous task, the other must be some recognizable sound frequency signal, but other audio signals are also suitable.

The `SHARC_Lab_LMS` project must be downloaded to the DSP board. Using the markings shown in Fig. 1, during the measurement the input signal $y(n)$ of the LMS structure must be connected to the left (marked in red) channel of the input of the signal processing card, and the reference signal $x(n)$ must be connected to the right (marked in white) channel. The signals $y(n)$, $\hat{y}(n)$, and $e(n)$ can be listened to and processed on the outputs of the card:

- y: Left and right channels of output 1
- $\hat{y}$: Left and right channels of output 2
- e: Left and right channels of output 3
- x: Left and right channels of output 4

1. Set up the adaptive echo reduction system, set the potentiometer from the position that achieves complete cancellation. Test the operation of the algorithm.
2. Try different values of μ and investigate the settling and stability properties of the system. How does changing μ affect the useful signal?

6.2.3 Active Noise Control (Optional)

Only one source is needed to solve the task.

The `SHARC_Lab_ANC` project must be downloaded to the DSP board. The program has two modes of operation: identification and noise reduction. In identification mode, the secondary path is identified, the white noise excitation is produced by the DSP board. If the error of the algorithm is sufficiently small, you can switch to noise reduction mode.

- Inputs:
  - Input 1 (white): reference signal
    in `ANC` mode: x
    in `Ident` mode: unused
  - Input 2 (red): error signal
    in `ANC` mode: e
    in `Ident` mode: y.
- Outputs:
  - Output 1 (both channels):
    in `ANC` mode: $-\hat{y}$
    in `Ident` mode: white noise
  - Output 2 (both channels):
    in `ANC` mode: error signal (e)
    in `Ident` mode: unused.

The identification mode can be activated by holding down the `FLAG1 / SW1` button. In identification mode, pseudo-random noise generated on the signal processing board is available on Output 1. In active noise reduction mode, by pressing the `FLAG2 / SW2` button, the coefficients of the $\hat{W}(z)$ filter can be reset, while the `DAI_P19 / SW3` button can be used to reset the coefficients of the $C(z)$ filter. By using the former, the settling properties of the system can be examined, while with the latter, the reference signal can be removed from the input of the LMS algorithm by filtering it with an all-zero coefficient filter. In the absence of a reference signal, the coefficients of $W(z)$ are not modified.

The output can be disabled using the `DAI_P20 / SW4` button. In order to preserve the stability of the filter coefficients, the adaptation of the coefficients of $W(z)$ is also suspended in case of a disabled output.

In this case, the noise signal must be applied to the `NOISE IN` input of the hybrid, which must be suppressed with the anti-noise signal applied to the `IN` input. The `OUT` output can be used to detect the error (the superposition of the noise and the result of the anti-noise). For this, the balancing of the hybrid must be eliminated.

1. Identify the secondary path. Based on the block diagram, determine what elements are in the signal path, and compare the result of the identification with this. (Use the graphic display of memory contents.)
2. Test the algorithm using a periodic noise signal.
3. What is the effect of changing the degrees of $W(z)$ and $C(z)$ and the value of μ on the degree of suppression, as well as on the settling and stability properties?