Digital implementation of a first-order analog low-pass filter

Fig.1 shows a low-pass filter built on the basis of discrete elements R and C and its counterpart represented by the impedance of these elements (1b).

The output signal of this filter can be determined using the current loop method (Kirchhoff's Voltage Law) or Laplace transfer function, treating the system as an impedance divider.

All signals should be treated as time dependent (u2(t), y(t) etc.). To simplify the analysis, initial conditions can be assumed to be zero.

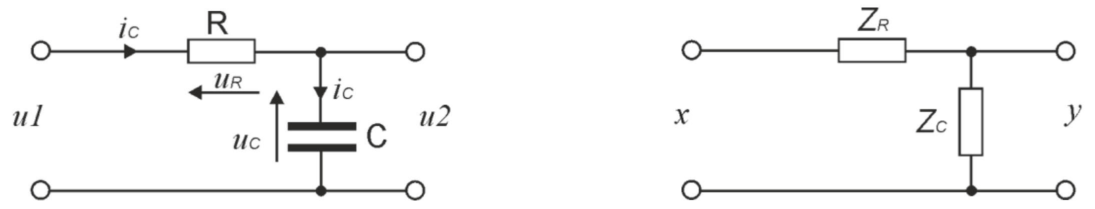


Fig.1. Low-pass analog filter of the first order.

a) analog RC implementation
b) impedance divider

$$u1 = u_R + u_C$$

$$u1 = i_C \cdot R + u_C$$

$$\begin{Bmatrix} u2 = u_C \\ i_C = C \cdot \frac{du_C}{dt} = C \cdot \frac{du2}{dt} \end{Bmatrix}$$

$$u1 = C \cdot \frac{du2}{dt} \cdot R + u2$$

$$\frac{du2}{dt} = (u1 - u2) \cdot \frac{1}{RC}$$

$$y = x \cdot \frac{Z_C}{Z_R + Z_C}$$

$$y = x \cdot \frac{\frac{1}{\omega \cdot C}}{R + \frac{1}{\omega \cdot C}}$$

$$y = x \cdot \frac{1}{1 + \omega \cdot RC}$$

$$\begin{Bmatrix} \omega \to s \\ s - Laplace\ operator \end{Bmatrix} \tag{1}$$

$$y = x \cdot \frac{1}{1 + s \cdot RC}$$

$$y \cdot s = (x - y) \cdot \frac{1}{RC}$$

$$\begin{Bmatrix} inverse\ \ Laplace \\ transform \end{Bmatrix}$$

$$\frac{dy}{dt} = (x - y) \cdot \frac{1}{RC}$$

In both cases we obtain identical differential equation for the derivative of the output signal in the form of:

$$\frac{dy}{dt} = (x - y) \cdot \frac{1}{T} \tag{2}$$

where T is the filter time constant:

$$T = R \cdot C \tag{3}$$

hence the 3-decibel cut-off frequency f_H is equal to:

$$f_H = \frac{1}{2 \cdot \pi \cdot RC} = \frac{1}{2 \cdot \pi \cdot T} \tag{4}$$

In order to digitally (discretely) calculate the filter output signal, the differential equation (2) should be transformed into a difference equation:

$$dy[n] = (x[n] - y[n]) \cdot \frac{1}{T} \tag{5}$$

Where:

n	- n’th step of data acquisition
dy[n]	- numerical calculation of the derivative of the output signal *dy/dt* – formula (2)
x[n]	- current value of the signal sample read from the ADC
y[n]	- the value of the filter output signal calculated at a given step

and then integrate.

There are many algorithms for numerical integration. They differ primarily in time consumption (number of operations) and accuracy of integral approximation. In real-time systems, the choice of method is a compromise between computational capabilities and required accuracy. The accuracy of calculations can be increased in two ways: by choosing a more accurate approximation (more complex algorithm) or by reducing the sampling time by increasing the sampling frequency.

The simplest approximation of the integral, called Euler's method, has the following form:

$$y[n] = y[n-1] + dy[n] \cdot \Delta t \tag{6}$$

$\Delta t = 1/f_s$ – sampling period

Substituting equation (5) into (6) we get:

$$y[n] = y[n-1] + (x[n] - y[n]) \cdot \frac{\Delta t}{T} \tag{7}$$

Transforming it further we get:

$$
\begin{aligned}
&y[n] = y[n-1] + x[n] \cdot \frac{\Delta t}{T} - y[n] \cdot \frac{\Delta t}{T} \\
&y[n] + y[n] \cdot \frac{\Delta t}{T} = x[n] \cdot \frac{\Delta t}{T} + y[n-1] \\
&y[n] \cdot (1 + \frac{\Delta t}{T}) = x[n] \cdot \frac{\Delta t}{T} + y[n-1] \\
&y[n] = x[n] \cdot \frac{\Delta t}{T+\Delta t} + y[n-1] \cdot \frac{\mathrm{T}}{T+\Delta t}
\end{aligned}
\tag{8}
$$

Substituting: $$\alpha = \frac{\Delta t}{T+\Delta t} \tag{9}$$

we obtain the final form of the signal value at the output of the digital low-pass filter:

$$\boxed{y[n] = x[n] \cdot \alpha + y[n-1] \cdot (1-\alpha)} \tag{10}$$

In turn, substituting: $$\beta = \frac{T+\Delta t}{\Delta t} = \frac{1}{\alpha} \tag{11}$$

we get a slightly different form of the filter. Since there is always true, that $\beta > 1$, this form makes it easier to implement the filter using integers:

$$\boxed{y[n] = \frac{x[n]+y[n-1]\cdot(\beta-1)}{\beta}} \tag{12}$$

The above implementation of a hardware digital filter, accessible to the user, can be found e.g. in Bosch sensors, BMP280 and others.

Using the equation (4) the filter cut-off frequency can be expressed using coefficients α or β:

$$f_C = \frac{1}{2\cdot\pi\cdot T} = \frac{\alpha}{2\cdot\pi\cdot\Delta t\cdot(1-\alpha)} = \frac{1}{2\cdot\pi\cdot\Delta t\cdot(\beta-1)} \tag{13}$$

However, the relation (10) is true only for $\alpha \leq 0.1$. Otherwise, the transfer characteristic of the digital filter differs from the characteristic of the analog filter for the same value of the time constant T and sampling frequency – it will be proved further.

One can try to determine these differences analytically, but it is easier to use numerical methods for this purpose because the final result always depends on the adopted method of approximating the integral, and therefore on the filter structure.

Let's determine the characteristics of the analog filter and the corresponding digital filter with the structure described by the equation (10).

We transform equation (10) in such a way that on the left side of the equation we group the terms with *'y'*, and on the right the terms with '*x'* to the form:

$$y[n] - y[n-1] \cdot (1-\alpha) = x[n] \cdot \alpha \tag{14}$$

We perform the **z**-transformation of the above equation on both sides and write it in the form of a transfer function:

$$Z\big[y[n]\big] - Z\big[y[n-1]\big] \cdot (1-\alpha) = Z\big[x[n]\big] \cdot \alpha$$

$$Y[z] - Y[z] \cdot z^{-1} \cdot (1-\alpha) = X[z] \cdot \alpha \qquad \big\{Z[y[n-k]] \to Y[z] \cdot z^{-k}\big\}$$

$$Y[z] \cdot (1 - z^{-1} \cdot (1-\alpha)) = X[z] \cdot \alpha$$

$$\frac{Y[z]}{X[z]} = \frac{\alpha}{1-z^{-1} \cdot (1-\alpha)} \tag{15}$$

Ignoring calculation errors resulting from different numerical representation of data (integers, floating-point numbers, rounding errors), the approximation error resulting only from the adopted filter structure can be estimated by comparing its amplitude-frequency characteristics for the adopted sampling period Δt with the characteristics of a continuous analog filter.

The amplitude-frequency (*a/f*) characteristics of an analog filter is obtained by substituting:

$$s \to j\omega \tag{16}$$

Hence the analog filter equation:

$$H_A(j \cdot \omega) = \frac{1}{1+j\omega \cdot T} \tag{17}$$

When making the substitution:

$$z^{-1} \to e^{-j\omega \cdot \Delta t} \tag{18}$$

we obtain the amplitude-frequency characteristic of the equivalent digital filter described by the equation (15):

$$H_D(j \cdot \omega) = \frac{\alpha}{1-(1-\alpha) \cdot e^{-j\omega \cdot \Delta t}} \tag{19}$$

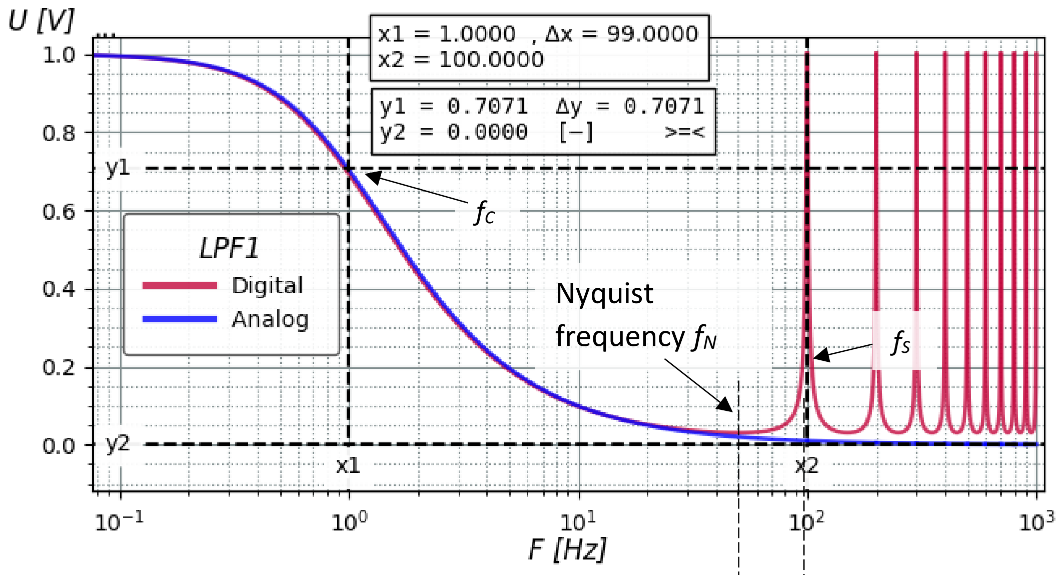
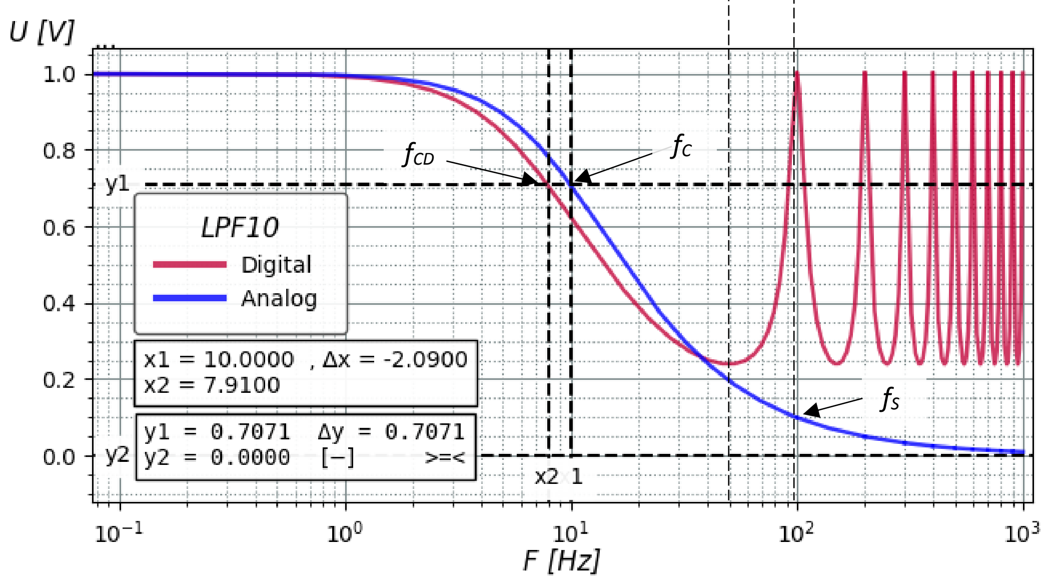


Fig.2. Low-pass filter band: f_C=1 Hz, f_S=100 Hz



Rys.3. Low-pass filter band: f_C=10 Hz, f_S=100 Hz

The above graphs show the characteristics of an analog low-pass filter with a cutoff frequency of 1 Hz (Fig. 2) and 10 Hz (Fig. 3), and the corresponding digital filter. In both cases, the signal sampling frequency is f_S=100Hz.

In the first case for f_C =1 Hz, $\alpha = \Delta t/(\Delta t + T) \approx 0.06$ the *a/f* characteristic of the digital filter coincides with the corresponding characteristic of the analog filter up to the Nyquist frequency ($f_N = f_S/2$).

In the second case for f_C =10 Hz, $\alpha \approx 0.39$, the *a/f* characteristic of the digital filter differs quite significantly from the characteristic of the analog filter. The cut-off frequency (crossing the level of *y1* in Fig. 3) in this case is $f_{CD} \approx 7.9$ Hz. With the increase of the cut-off frequency of the analog filter on which the digital filter is modeled, these differences increase, but the cut-off frequency of the digital filter is always lower.

The above example clearly shows that in the case of digital filtering that imitates analog, an important factor is the ratio of the signal sampling frequency to the filter cutoff frequency. In other words, the higher this ratio is, the better the digital filter emulates the analog one. As it decreases, the digital filter bandwidth narrows. In addition, this frequency criterion is much more stricter than the Nyquist criterion - in the first case it is as much as 100, and in the second it is equal to 10.

Of course, it is possible to design a digital filter emulating an analog filter for a low sampling frequency, but other values of the α constant should be assumed than those resulting from the analog filter parameters. For the discussed case (f_C=10Hz, f_S=100Hz), $\alpha \approx 0.46$ should be assumed.

Another solution is to use higher order filters or, as mentioned earlier, another integral approximation algorithm. In practice, cascades of second order filters are used. A commonly used integral approximation algorithm is the trapezoidal approximation method (other names are the bilinear/Tustin transform), but this is not a rigid rule.

For example, applying the Tustin transformation to filter (1) requires the substitution:

$$s \rightarrow \frac{2}{\Delta t} \cdot \frac{1-z^{-1}}{1+z^{-1}} \tag{20}$$

As a result, we get the transfer function of the digital filter in the form:

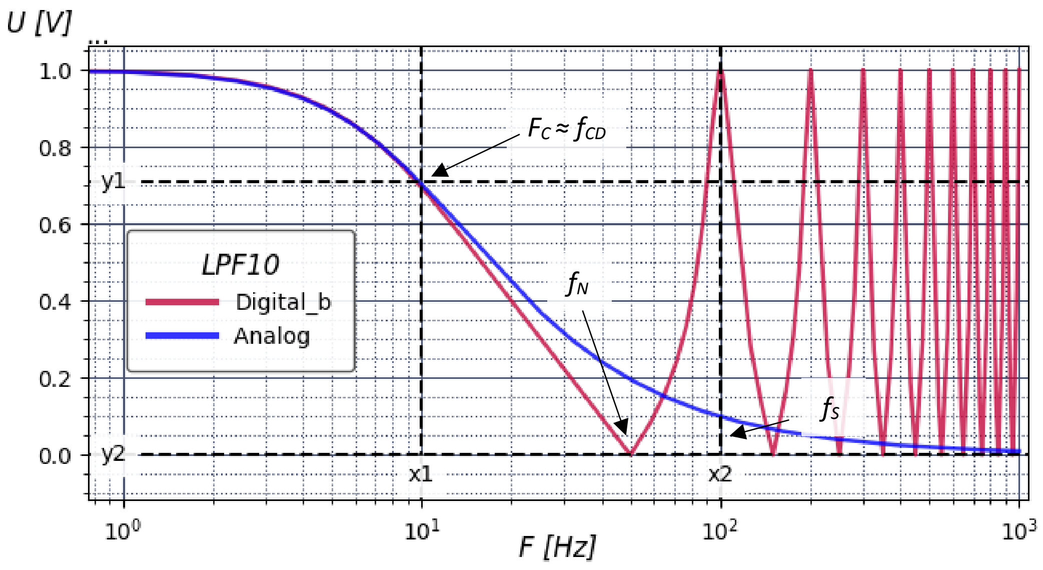
$$H_T[z] = \frac{Y[z]}{X[z]} = \frac{B_0+B_1 \cdot z^{-1}}{1+A_1 \cdot z^{-1}} \tag{21}$$

This filter implementation, in the simplest case, gives the following difference equation:

$$y[n] = B0 \cdot x[n] + B1 \cdot x[n-1] - A1 \cdot y[n-1] \tag{22}$$

$$\left\{ Z^*[z^{-k}] \rightarrow x[n-k] \right\}$$

Compared to (10), it requires additional arithmetic operations and storing the previous value of the input signal *x[n-1]*.



Rys.4. Low-pass filter band: f_C=10 Hz, f_S=100 Hz
Approximation of the integral by the trapezoidal method.

In this case, the cut-off frequency of the digital filter coincides with the cut-off frequency of the analog filter, and in the higher frequency range it has even better attenuation.

The presented *a/f* characteristics of the digital filter show that, unlike in the analog filter, its bandwidth above the Nyquist frequency does not decay to zero but is infinitely periodic. Its impulse response is also infinite. Hence the name of such a filter – Infinite Impulse Response Filter – IIR Filter.

The term "low-pass" in this case refers to the parameters of the analog filter, which was the reference point for the digital filter and such a selection of the sampling frequency so that the input signal bandwidth is smaller than the Nyquist frequency. If for a given sampling frequency we do not limit the signal bandwidth in this way before the ADC converter, the digital filter will not limit it either. It will pass according to its bandwidth. The bandwidth limitation may result from the signal source, e.g. measurement of physical quantities with a bandwidth limited due to the physical nature of the object or it must be made using a real analog filter.

This document provides a simple analysis useful for designing digital low-pass filters for use in microprocessor data acquisition systems.

The calculations were performed using the *signal* module from the *SciPy* Python library and the plots were generated using the *Matplotlib* module

A sample Python program for determining the amplitude-phase-frequency characteristics is included in a separate document.

SUPPLEMENT

Using filter coefficients in software implementation.

In the literature it is common to represent the transfer function of analog filters using the Laplace transform. The general equation of the transfer function has the following form:

$$H[s] = \frac{b_M \cdot s^0 + b_{M-1} \cdot s^1 + b_{M-2} \cdot s^2 + \cdots + b_0 \cdot s^M}{1 + a_{N-1} \cdot s^1 + a_{N-2} \cdot s^2 + \cdots a_0 \cdot s^N} \tag{23}$$

The order of the filter is defined here by the value of the largest exponent standing for the variable $\boldsymbol{s}$ in the denominator, i.e. N. This is a normed form where $a_0 = 1$ i $M \leq N$. Practically, the description of each filter can be reduced to this form.

For the analog filter described by equations (1), the coefficients a and b have the following values:

$b_0 = 1$, $a_0 = \mathrm{T}$, $a_1 = 1$

The previously presented method of analytical determination of the digital filter equation is time-consuming for more complicated filters. Computer programs such as *Matlab* or *SciPy* (Python) use the coefficients a and b to determine the coefficients A and B of the $\boldsymbol{z}$-transform of the digital filter.

The general form of the digital filter transfer function presented by the $\boldsymbol{z}$-transform is:

$$\frac{Y[z]}{X[z]} = \frac{B_0 + B_1 \cdot z^{-1} + B_2 \cdot z^{-2} + \cdots + B_M \cdot z^{-M}}{1 + A_1 \cdot z^{-1} + A_1 \cdot z^{-2} + \cdots + A_N \cdot z^{-N}} \tag{24}$$

For the digital filters described by equations (10, 15) oraz (21, 22), dla f_H=10 Hz, f_S=100 Hz, the coefficients are as follows:

Euler's method:

$B_0 = alfa = 0.385869545$ $B_1 = 0$

$A_0 = 1$ $A_1 = -(1 - alfa) = -0.614130455$

bilinear/Tustin's transformation:

$B_0 = 0.23905722$, $B_1 = 0.23905722$

$A_0 = 1$ $A_1 = -0.52188555$

In the general case, the difference equation to be implemented in the microprocessor program, determined on the basis of the calculated coefficients *A* and *B*, for the second-order filter, will have the following form (inverse $\boldsymbol{z}$-transform of equation (24)):

$$\begin{aligned} y[n] = B0 \cdot x[n] + B1 \cdot x[n-1] + B2 \cdot x[n-2] + \\ -A1 \cdot y[n-1] - A2 \cdot y[n-2] \end{aligned} \quad (25)$$

$$\left\{ Z^{*}[z^{-k}] \rightarrow x[n-k] \right\}$$

There is always *A0* = 1 and it does not appear in the calculations any more. This form of the output equation is called *Direct Form 1 - DF1.*

The software implementation of equation (25) requires storing two previous samples of the input signal *x[n-1]* and *x[n-2],* and the previous results of the calculated filter output signal *y[n-1]* i *y[n-2].* This is not an optimal form due to the speed of calculations.

Equation (24) can be transformed as follows:

$$\frac{Y[z]}{X[z]} = \frac{N[z]}{D[z]} \quad (26)$$

$$Y[z] = \frac{N[z]}{D[z]} \cdot X[z] = \frac{X[z]}{D[z]} \cdot N[z] = W[z] \cdot N[z] \quad (27)$$

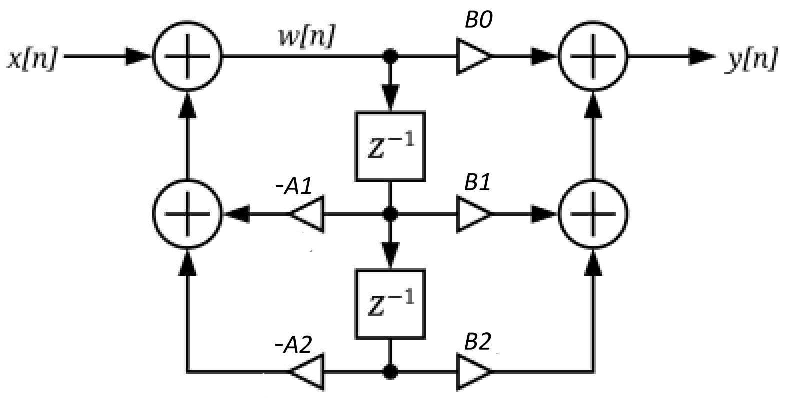
Then the difference equation will have the following form:

$$\begin{aligned} w[n] &= x[n] - A1 \cdot w[n-1] - A2 \cdot w[n-2] \\ y[n] &= B0 \cdot w[n] + B1 \cdot w[n-1] + B2 \cdot w[n-2] \end{aligned} \quad (28)$$

This implementation is called *Direct Form 2 - DF2.* It no longer requires storing the previous samples of the input and output signal, but only the previous two samples of the auxiliary variable $\boldsymbol{w}$. In analogy to control theory, it can be treated as a filter state variable.

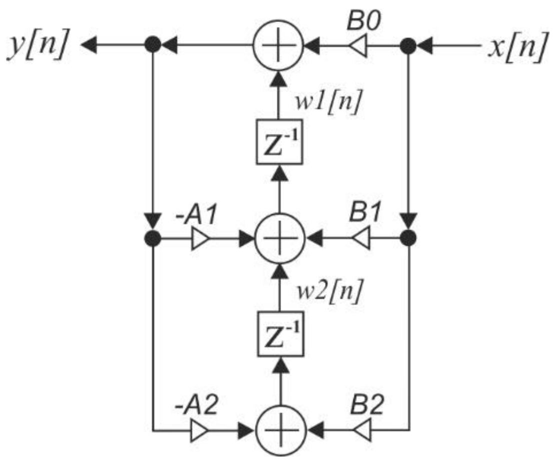
Another form of difference equations can be obtained from both *DF1* and *DF2* formulas by exchanging the filter input with its output and appropriately transforming the signal flow. Such an exchange does not change the filter transfer function. This is best shown by the example of transforming the signal flow graph in the filter.

On the example of the *DF2* form:



Rys.5. Signal flow graph of the *DF2* implementation of the digital IIR filter.
The $\boldsymbol{z^{-1}}$ component in the diagram represents a delay of one sampling period.

By swapping the input with the output, the above diagram must be transformed according to the principle: the directions of flow of all signal paths are reversed, branches are transformed into sums, and sums into branches. In this way, we obtain *Direct Form 2 Transposed – DF2T*.



Rys.6. Signal flow graph of the *DF2T* implementation of the digital IIR filter.

Based on the above functional diagram, the system of equations can be written as follows:

$$\begin{aligned} y[n] &= w1[n] + B0 \cdot x[n] \\ w1[n+1] &= w2[n] + B1 \cdot x[n] - A1 \cdot y[n] \\ w2[n+1] &= B2 \cdot x[n] - A2 \cdot y[n] \end{aligned} \tag{29}$$

The calculations should be performed in the order indicated above. In this implementation, two variables *w1* and *w2* should also be stored. The index *[n+1]* means that the values calculated in step *[n]* will be used only in the next step *[n+1]*.

Theoretically, all of the above implementations are equivalent. However, in practice, the choice depends on the specific implementation, the type of filter, its frequency characteristics, the choice of representation and precision of numbers, as well as the target element in which this filter will be implemented: a microprocessor calculating the filter in a purely programmatic way or an FPGA with specialized components for implementing such a filter. The *DF1* form is usually used for simple filters as in the example given at the beginning. In more complex filters, the remaining forms are used. The simplest way to verify the correct operation of a digital filter is a numerical simulation taking into account all aspects of the filter's operation in the target conditions. A detailed analysis of the correct choice of the form of realization of a digital filter is beyond the scope of this study.