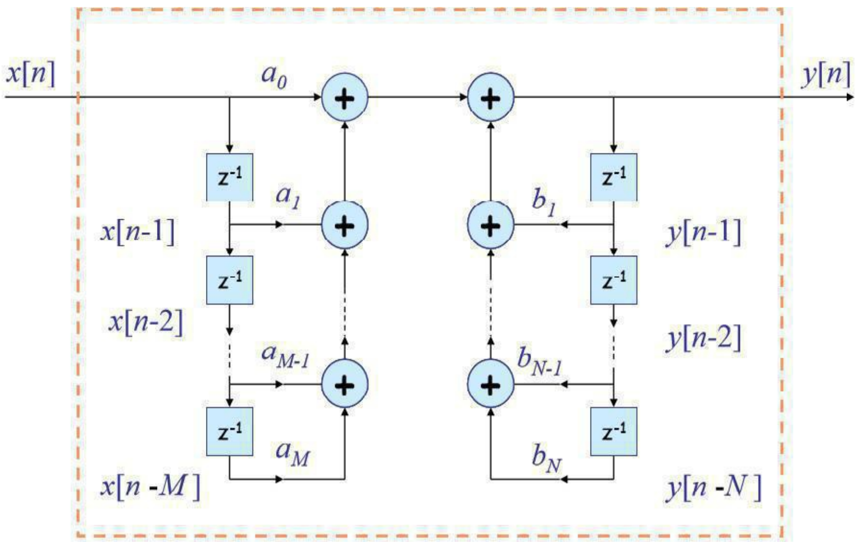
IIR filter implementation

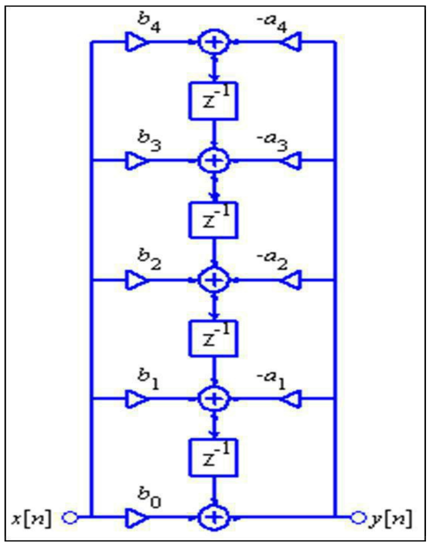
IIR filters are used in many digital signal processing applications despite the fact they do not have a linear phase. The main advantage compared to FIR filters is in the fact to have a smaller number of coefficients to achieve the same filtering performance in the frequency domain. IIR filters can be represented using the difference equation, and the corresponding transfer function in the z domain.

$$y(n) = \sum_{i=0}^{N} (b_i x(n-i)) - \sum_{i=1}^{N} (a_i y(n-i))$$

$$H(z) = \frac{b_0 + b_1 z^{-1} + \ldots + b_N z^{-N}}{1 + a_1 z^{-1} + \ldots + a_N z^{-N}}$$

Based on these two equations we define the two implementations forms for IIR filter. These are the direct form I, and the direct form II both transposed.





$$B = [b_0\ b_1\ b_2\ b_3\ b_4]$$

$$A = [1\ a_1\ a_2\ a_3\ a_4]$$

Filter design means that we need to determine the filter coefficients for arrays B and A in order to fulfill the filtering requirements. We assume that the filter coefficients are obtained, for example by using the fourth order Chebyshev II approximation. Coefficients are:

Numerator

b4 =9.733517399250140E-3

b3= 1.871580508766260E-2

b2 =2.484970675674230E-2

b1= 1.871580508766260E-2

b0= 9.733517399250140E-3Denominator

a4= 1.852704707109540E-1

a3= -1.044283378662510E0
a2= 2.298169395109720E0
a1= -2.357408135427590E0
a0= 1.000000000000000E0

We define the vectors of coefficients using the values from above, in order to illustrate how the IIR filter is used to perform the filtering of a signal.

$N := 4$ IIR filter order

$$NumCoeff := \begin{bmatrix} 9.7335174 \\ 18.715805 \\ 24.849707 \\ 18.715805 \\ 9.7335174 \end{bmatrix} \cdot 0.001$$ IIR filter numerator coefficients

$$DenomCoeff := \begin{bmatrix} 1.0 \\ -2.3574081 \\ 2.2981694 \\ -1.0442834 \\ 0.1852704 \end{bmatrix}$$ IIR filter denominator coefficients

$Fs := 20000$ Hz, sampling frequency

$Ts1 := 1 / Fs$ s, sampling period

$dt := ynodes(x, 0, 0.005 - Ts1, 100)$ Time variable

$f1 := 1000$ Hz, frequency

$f2 := 6000$ Hz, frequency

$x := 5\sin(2\pi \cdot f1\ dt) + 5\sin(2\pi \cdot f2\ dt)$ Test signal, sum of two sinusoidals

$NumSigPoints := size(x)$ Length of the test signal

The test signal has been defined above, and we can call the IIR filtering function to attenuate the sinusoidal component of the higher frequency. First, we must determine the roots of the denominator in order to check whether the poles of the filter are within the unit circle, which imposes the stability of the IIR filter. After that, the IIR filter is initialized and the test signal is filtered by using the function iirfilter() whose arguments are signal and filter coefficients.

$$fabs(polroots(coef2expr(DenomCoeff))) = \begin{bmatrix} 0.536 \\ 0.536 \\ 0.803 \\ 0.803 \end{bmatrix}$$ Poles of the IIR filter

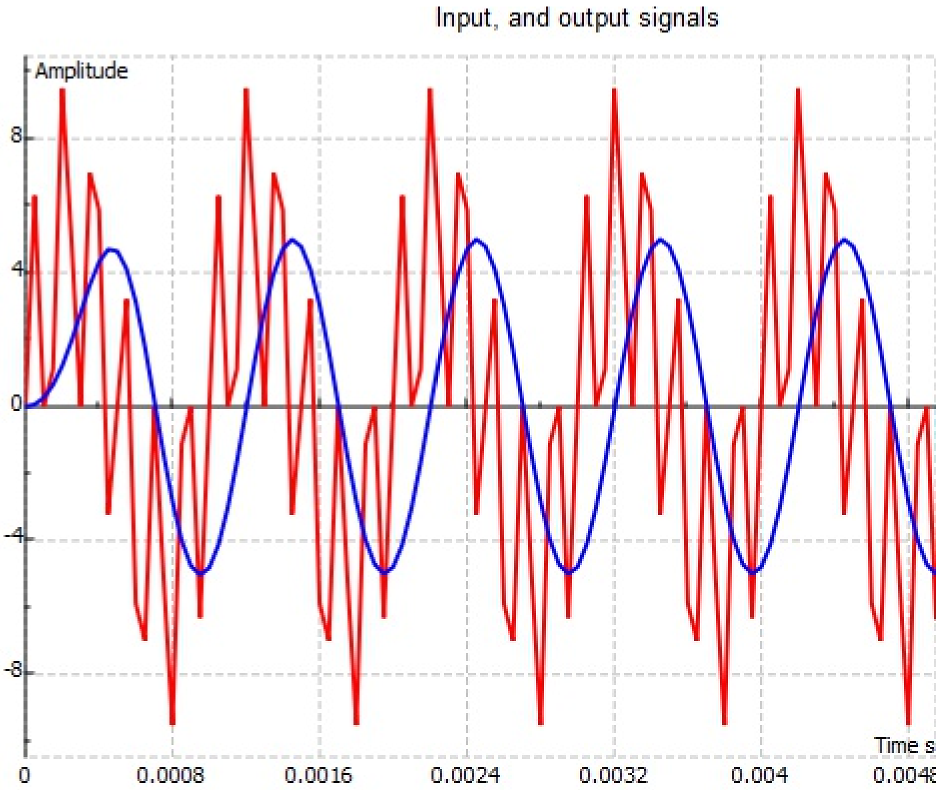
$initiirfilter(0, size(DenomCoeff))$ Initialize IIR filter of required order

$y := iirfilter(0, x, DenomCoeff, NumCoeff)$ Filter the test signal using initialized filter

We can show the input signal, and filtered signal in the same graph. It is obvious that the sinusoidal of higher frequency has been attenuated.

graf1 := join mat cols(dt , x)   Graph of the input test signal

graf1f := join mat cols(dt , y)   Graph of the output filtered signal



Analysis of the IIR filter

In a sequel we analyze the IIR filter in the frequency domain in order to see its performance. The filter analysis means that we determine its frequency response, which can be visualized as an amplitude and phase response. Sometimes, there it is needed to determine the phase delay and group delay.

The filter frequency response in the desired number of points is given in the next graph.

H1 := iirfreqres(DenomCoeff , NumCoeff , 128 , 1)   Calculate frequency response of the filter

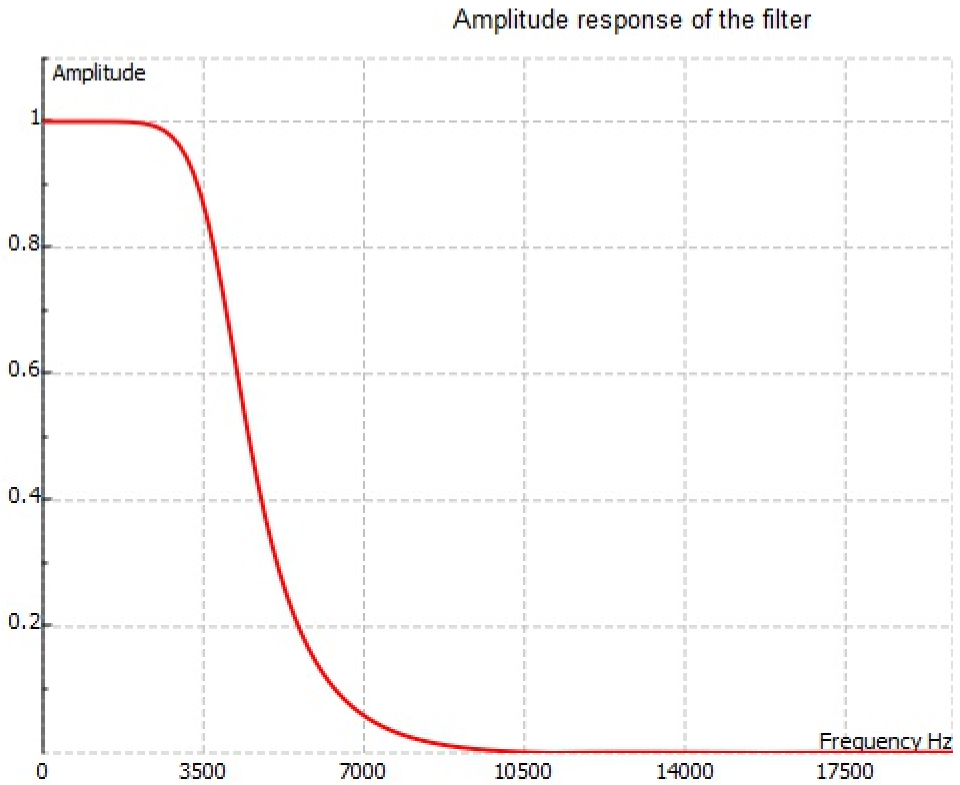
Habs := fabs(H1)   Amplitude response is absolute value of the frequency response

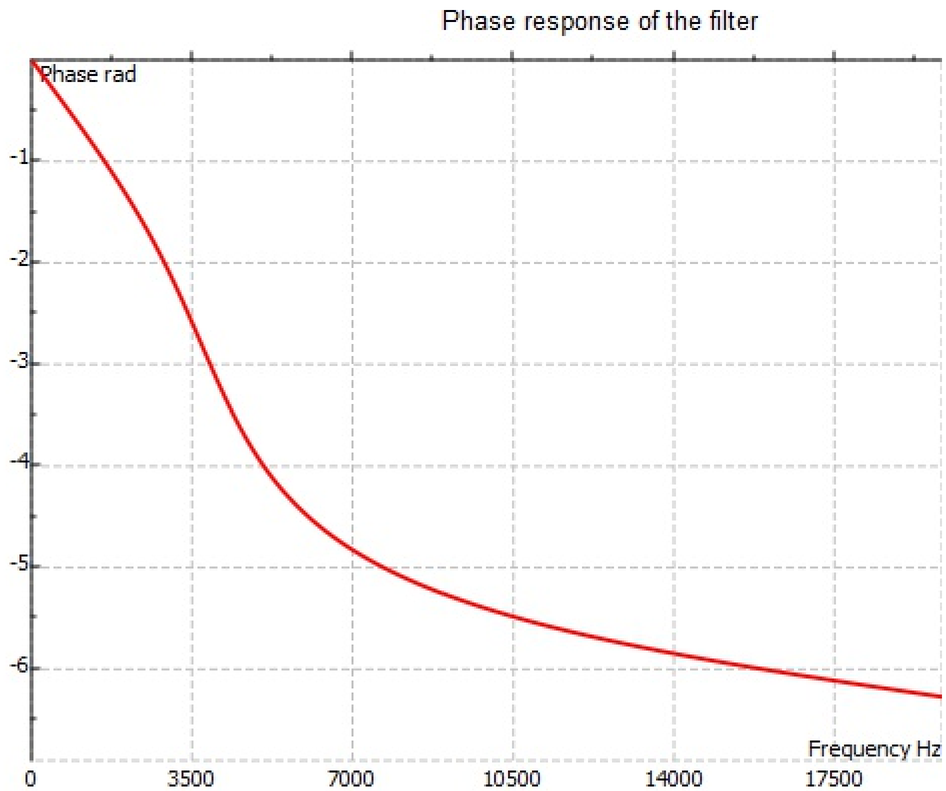
fre := ynodes(z , 0 , 1 - 1/128 , 128)   Frequency axis

graf2 := join mat cols(fre · Fs , Habs)   Graph of the amplitude response

Next, we determine the phase response of the filter.

graf3 := join mat cols(fre · Fs , contphase(H1))   Phase response of the filter

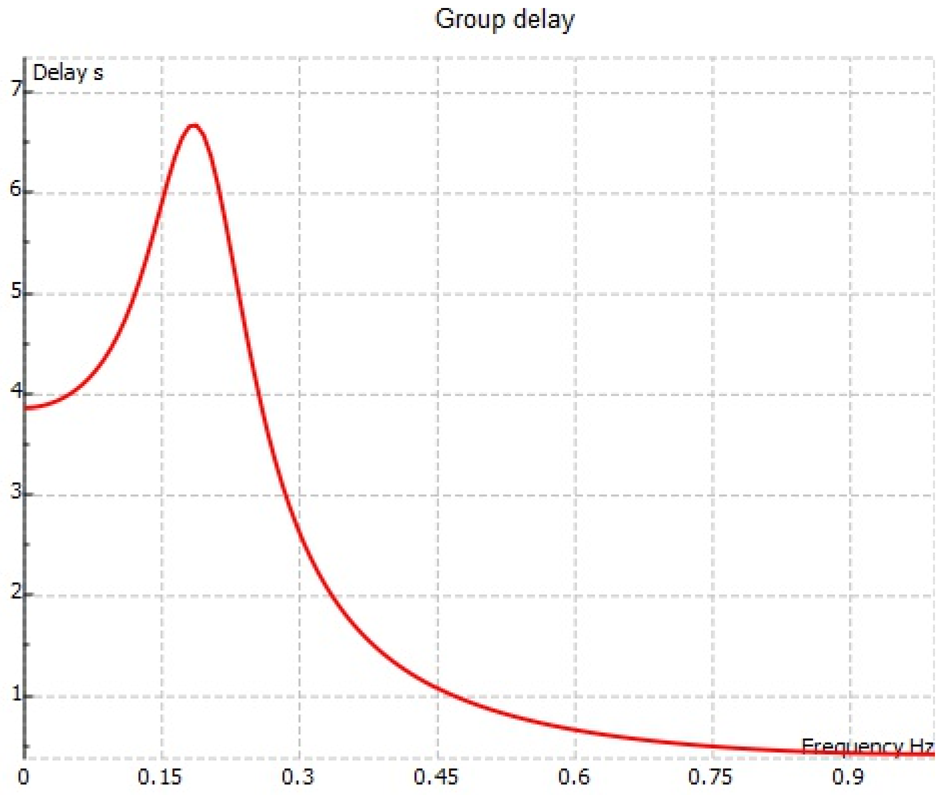




In the next part we generate a function to calculate the group delay of an IIR filter. Here, we use the function iirgroupdelay() made in script language within this document. MatDeck's function for the calculation of group delay with the same purpose is iirgrpdelay(), it is used in the same manner.

grpd := iirgroupdelay(DenomCoeff , NumCoeff , 128)

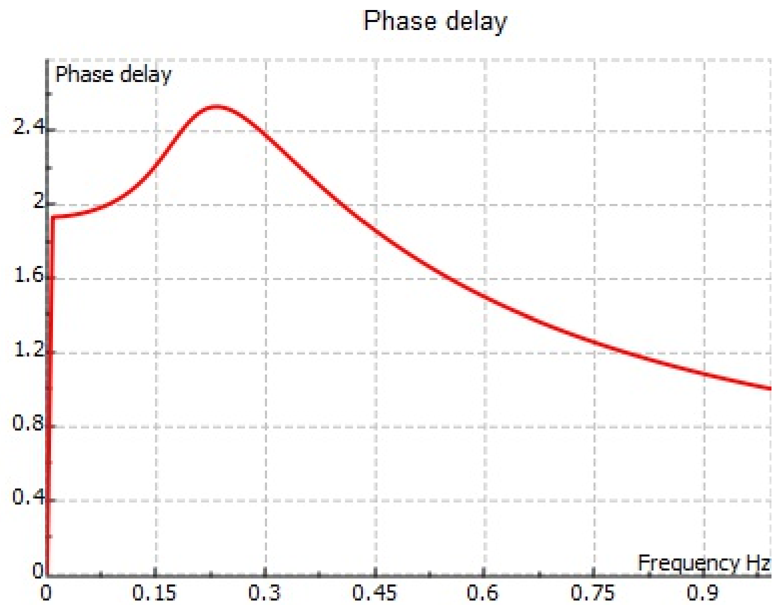
grpdgraf := join mat cols(fre , grpd) Graph of the group delay



In the same manner, the phase delay of the filter can be calculated. The phase delay is shown in the next graph.

phd := iirphasedelay(DenomCoeff , NumCoeff , 128)

phdgraf := join mat cols(fre , phd) Graph of the phase delay



```
iirphasedelay(vec1 , vec2 , numpoints)
{
    frres := iirfreqres(vec1 , vec2 , numpoints , 1)
    phres := contphase(frres)
    fr := vector create(numpoints , 0 , 0)
    mat := vector create(numpoints , 0 , 0)
    for(i := 1 , i < numpoints , i += 1 )
    {
        fr[i] = (2 · π) · i / numpoints
        phres[i] = 0 - phres[i] / fr[i]
    }

    return(phres)
}
```

Code of the function to calculate phase delay

```
iirgroupdelay(vec1 , vec2 , numpoints)
{
    ve1 := vec1
    ve2 := vec2
    oa := size(ve1) - 1
    ob := size(ve2) - 1
    if(oa < 0 )
    {
        ve1 = 1
        oa = 0
    }
    if(ob < 0 )
    {
        ve2 = 1
        ob = 0
    }
    oc := oa + ob
    c := convolution(ve2 , flip(conj(ve1) , 1))
    cr := vector create(oc + 1 , 0 , 0)
    for(i := 0 , i <= oc , i += 1 )
    {
        cr[i] = c[i] · i
    }
    num := fft1n(cr , 2 numpoints)
    den := fft1n(c , 2 numpoints)
    Matr := vector create(numpoints , 0 , 0)
    for(j := 0 , j < numpoints , j += 1 )
    {
        Matr[j] = complexre(num[j] / den[j]) - oa
    }
    return(Matr)
}
```

Code of the function which calculates the group delay