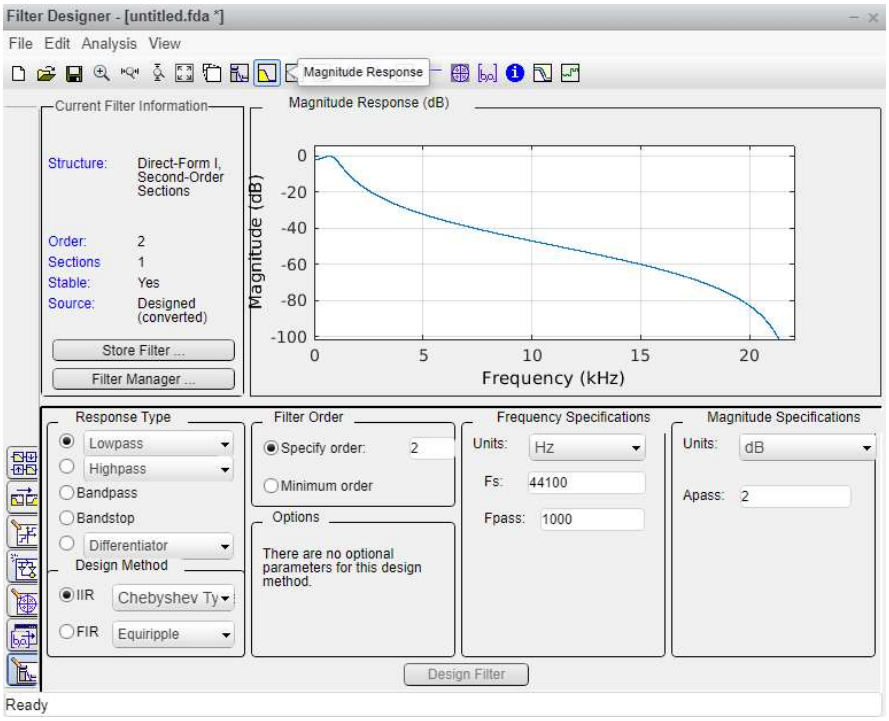
## 3.2. IIR filtrai

### Žemų dažnių IIR filtras

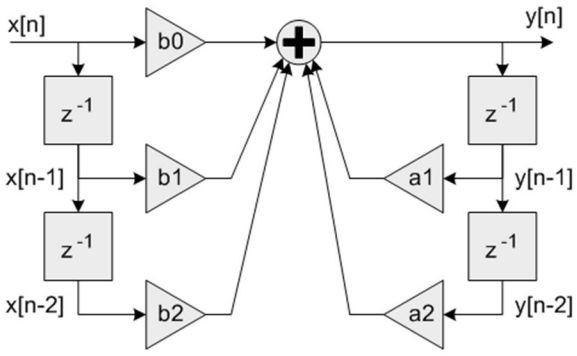
Žemų dažnių IIR filtro funkcijos pavyzdys, nenaudojant CMSIS DSP bibliotekų. Filtro koeficientai randami MATLAB filterDesigner įrankio pagalba.




1 pav. MATLAB filterDesigner

Eksportuoti filtro koeficientai:

```
#define NUM_SECTIONS 1

float b[NUM_SECTIONS][3] = {
{3.94806912E-03, 7.89613824E-03, 3.94806912E-03} };

float a[NUM_SECTIONS][3] = {
{1.00000000E+00, -1.87614073E+00, 8.91933002E-01} };
```




2 pav. "*Biquad Direct form I*" filtro sekcijos struktūra

IIR filtro lygtis:

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

arba

```
y[n] = b0 * x[n] + b1 * x[n-1] + b2 * x[n-2] - a1 * y[n-1] - a2 * y[n-2]
```

Filtro funkcija, realizuota C kalba:

```
//LP filter
            int16_t LP_filter(int16_t InSample)
            {
                        float_t InSampleF = (float_t)InSample;

                        float_t OutSampleF =
                                                lp_b0 * InSampleF
                                                + lp_b1 * lp_z1_in
                                                + lp_b2 * lp_z2_in
                                                - lp_a1 * lp_z1_out
                                                - lp_a2 * lp_z2_out;

                        lp_z2_in = lp_z1_in;
                        lp_z1_in = InSampleF;
                        lp_z2_out = lp_z1_out;
                        lp_z1_out = OutSampleF;

                        return (int16_t)OutSampleF;
            }
```

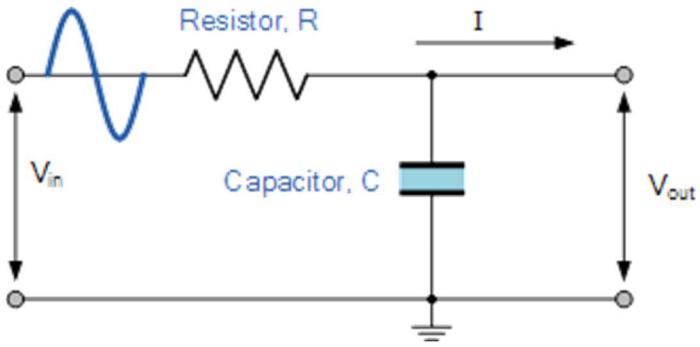
## IIR filtrų fazės netiesiškumas

Nesunku padaryti, kad FIR filtras turėtų tiesinę fazę. Taip yra todėl, nes impulsinis atsakas (filtro branduolys) yra tiesiogiai sudarytas filtro projektavimo metu. Viskas, ko reikia – tai, kad filtro branduolys būtų simetriškas iš kairės į dešinę.

Taip nėra IIR filtrų atveju, nes skaičiavimuose naudojami rekursijos koeficientai, o ne impulsinis atsakas. Rekursinio filtro impulsinis atsakas nėra simetriškas tarp kairės ir dešinės, todėl turi netiesinę fazę.

### Žemo dažnio analoginis RC filtras

Paprastas pasyvus žemo dažnio RC filtras gali būti nesunkiai įgyvendintas nuosekliai sujungus rezistorių su kondensatoriumi, kaip parodyta pav. Šiuo atveju įėjimo signalas (įtampa *Vin*) paduodama tarp rezistoriaus ir kondensatoriaus grandinės, o išėjimo signalas (įtampa *Vout*) nuskaitoma „ant“ kondensatoriaus.




Tokio tipo filtras dar vadinamas 1-os eilės arba vieno poliaus filtru, kadangi turi tik vieną reaktyvųjį elementą – kondensatorių.

Kondensatoriaus reaktyvioji varža priklausys nuo signalo dažnio – esant žemam dažniui bus didelė, ir atvirkščiai. Tai reiškia, kad žemame dažnyje kondensatoriaus varža bus daug didesnė už rezistoriaus, todėl įtampos kritimas bus daug didesnis už $V_R$. Tuo tarpu esant aukštiems dažniams kondensatoriaus varža mažės, įtampos kritimas *Vc* taip pat bus mažas ir aukštesnių dažnių signalas bus atitinkamai slopinamas daug labiau.

Grandinės išėjimo įtampą galima apskaičiuoti:

$$V_{out} = V_{in} \times \frac{R_2}{R_1 + R_2}$$

arba

$$V_{out} = V_{in} \times \frac{X_C}{Z}$$

čia *Xc* - kondensatoriaus reaktyvioje varža:

$$X_C = \frac{1}{2\pi f C}$$

*Z* - grandinės impedansas:

$$Z = \sqrt{R^2 + X_C^2}$$

*Skaičiavimų pavyzdys:* filtro grandinė sudaryta iš 4,7kΩ rezistoriaus ir 47nF kondensatoriaus. Įėjimo signalas – 10V sinusinis signalas. Kokia išėjimo įtampa, esant 100Hz ir 10kHz signalo dažniui?

1. Išėjimo įtampa *Vout*, esant 100Hz įėjimo signalui

$$X_C = \frac{1}{2\pi fC} = \frac{1}{2\pi \times 10 \times 47 \times 10^{-9}} = 33.863\Omega$$

$$V_{out} = 10 \times \frac{33863}{\sqrt{4700^2 + 33863^2}} = 9.9V$$

2. Išėjimo įtampa *Vout*, esant 10kHz

$$X_C = \frac{1}{2\pi fC} = \frac{1}{2\pi \times 10000 \times 47 \times 10^{-9}} = 338.6\Omega$$

$$V_{out} = 10 \times \frac{338.6}{\sqrt{4700^2 + 338.6^2}} = 0.718V$$

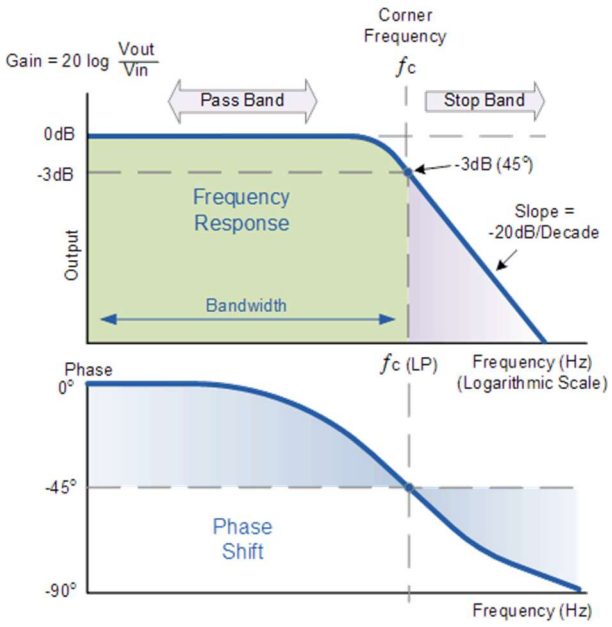
3. Filtro atkirtos dažnis *fc*

$$f_c = \frac{1}{2\pi RC} = \frac{1}{2\pi \times 4700 \times 47 \times 10^{-9}} = 720Hz$$

4. Fazės poslinkis φ

$$Fazės\ poslinkis\ \varphi = -\arctan(2\pi fRC)$$

1-os eilės RC filtro dažninis atsakas parodytas pav. žemiau.




Pralaidos zonoje (*pass band*) filtras praleidžia visus dažnius su nedideliu slopinimu (stiprinimo koeficientas artimas 1), kol pasiekiamas atkirtos dažnis *fc*.

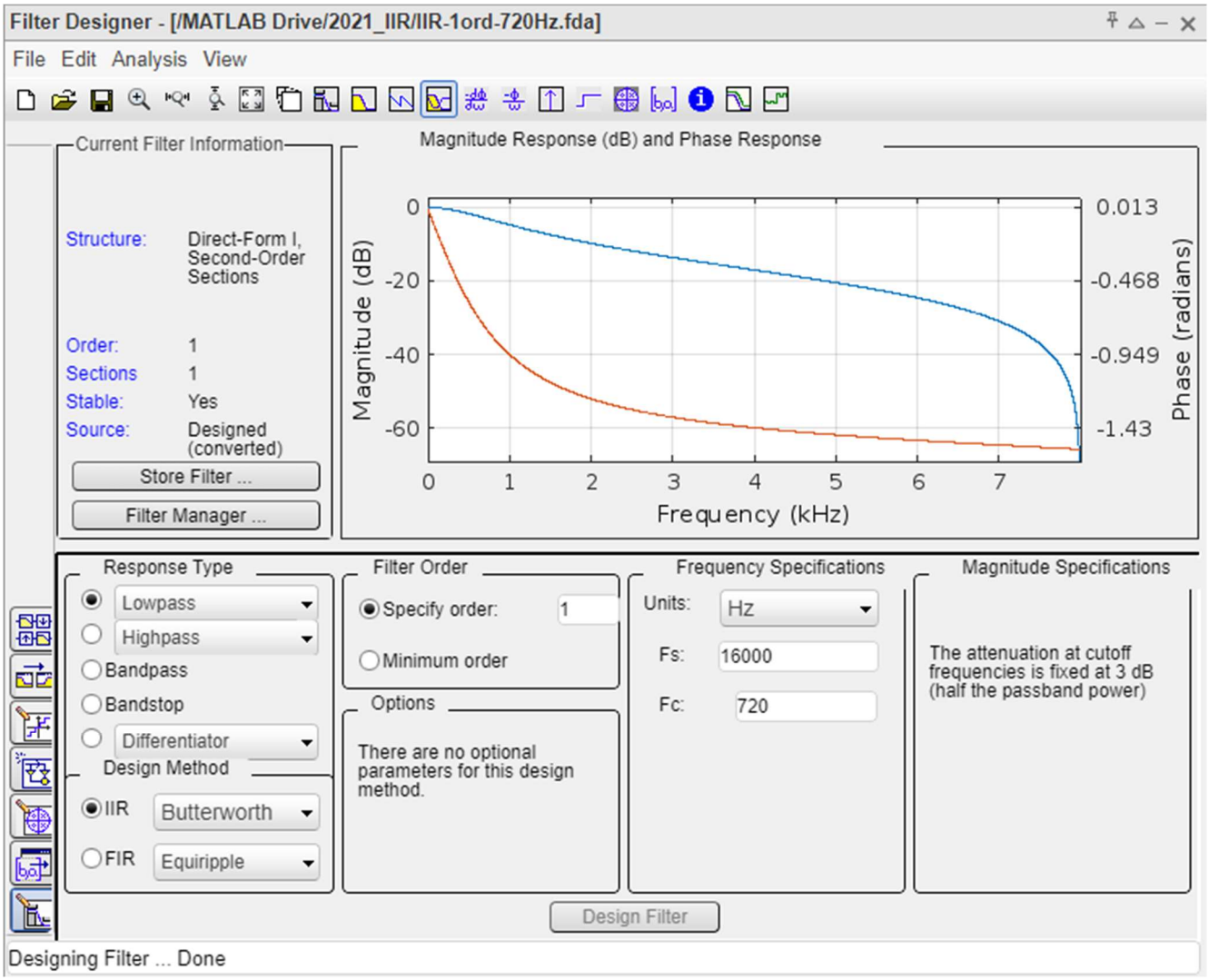
Nuo šio atkirtos dažnio grandinės atsakas žemėja iki nulio -20dB/dekadai sparta (*dekada – dažnio pokytis 10 kartų*).

Atkirtos dažnis nustatomas kaip dažnis, ties kuriuo rezistoriaus aktyvioji ir kondensatoriaus reaktyvioji varžos susilygina, t.y. $R = Xc = 4.7k\Omega$. Ties šiuo dažniu išėjimo signalas yra lygus 70,7% įėjimo signalo arba -3dB.

Kadangi grandinėje naudojamas kondensatorius, išėjimo signalas „atsilieka“ nuo įėjimo signalo ir ties atkirtos dažniu yra -45º. Kuo didesnis įėjimo signalo dažnis, tuo didesnis bus fazių skirtumas.

**Žemo dažnio skaitmeninis IIR filtras**

MatLab FilterDesigner įrankio pagalba suprojektuotas žemų dažnių filtras su 720Hz atkirtos dažniu:




Eksportuoti skaitmeninio filtro koeficientai:

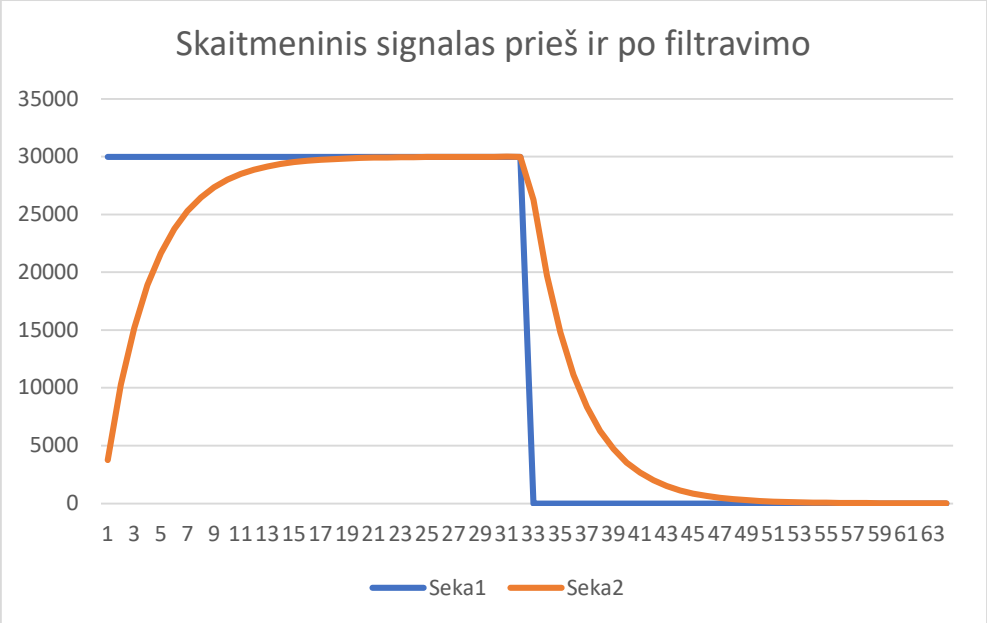
```
#define NUM_SECTIONS 1

float b[NUM_SECTIONS][3] = {
{1.24589381E-01, 1.24589381E-01, 0.00000000E+00} };

float a[NUM_SECTIONS][3] = {
{1.00000000E+00, -7.50821238E-01, 0.00000000E+00} };
```

Atliekamas stačiakampio 255 Hz dažnio signalo filtravimas CMSIS-DSP bibliotekų pagalba. Žalia spalva (B kanalas) audio-codec'u atkurtas signalas prieš filtravimą, raudona spalva (A kanalas) – po skaitmeninio filtro.

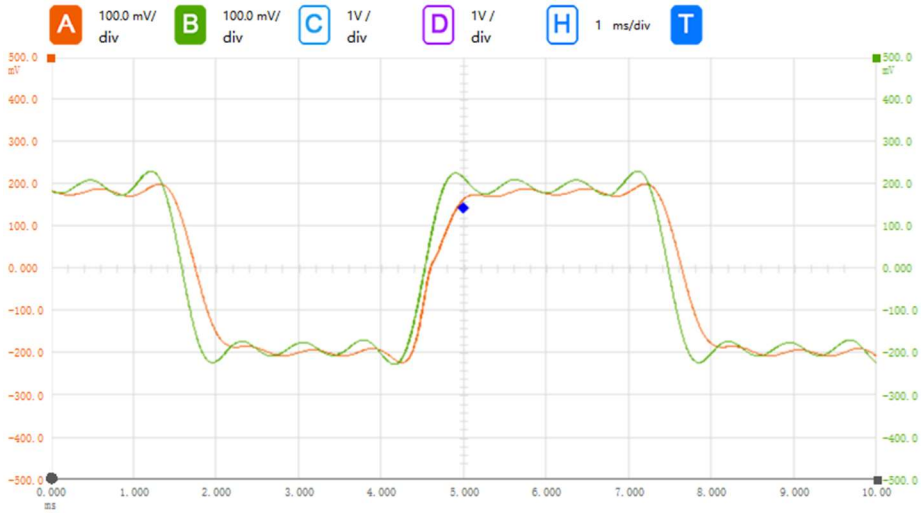

Result

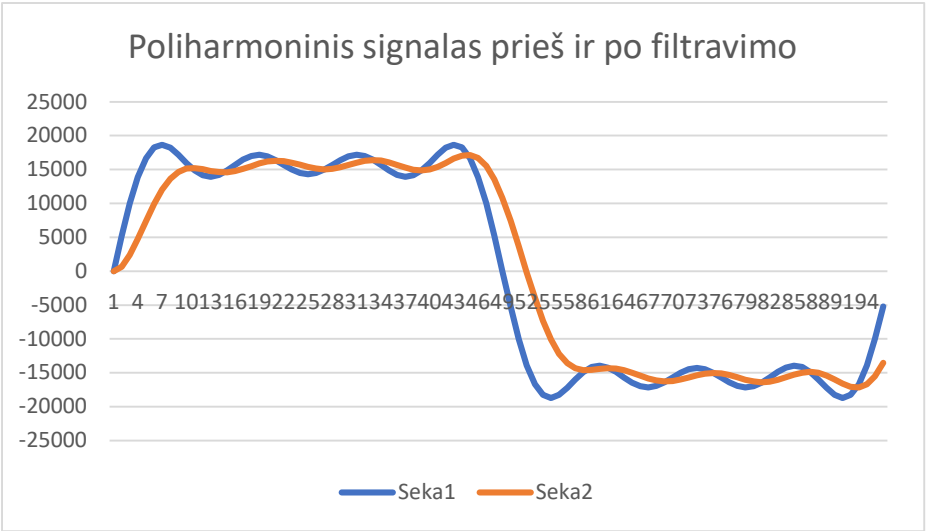
| Channel | Name | Span | Value | Minimum | Maximum | Average | σ |
|---|---|---|---|---|---|---|---|
| A | Frequency | Whole Trace | 254.777 Hz | 254.715 Hz | 254.819 Hz | 254.780 Hz | 0.024 Hz |

Skaitmeninis signalas prieš ir po filtravimą – įėjimo signalo ir filtruoto signalų buferių reikšmės MV atmintyje:



Poli-harmoninio signalo filtravimo pavyzdys (signalo dažnis – 170 Hz, harmonikų kiekis – 4):




**„Zero phase“ filtras**

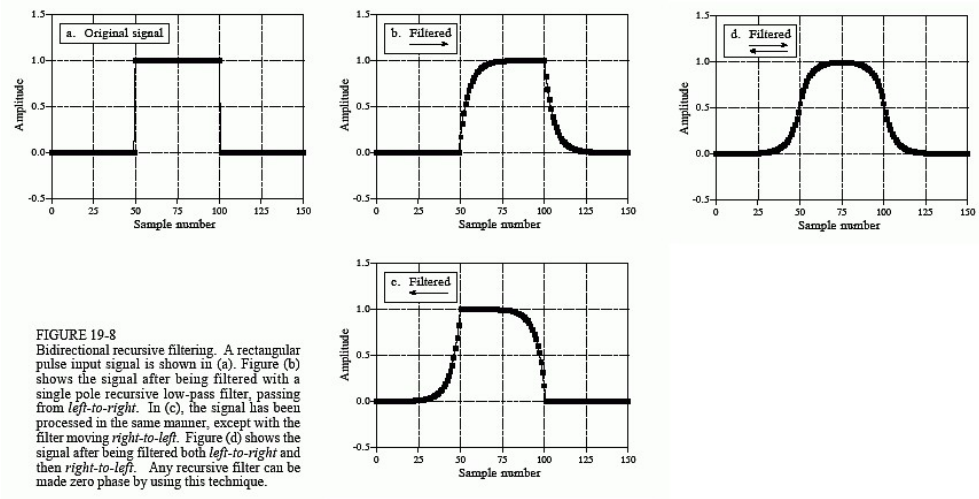



FIGURE 19-8
Bidirectional recursive filtering. A rectangular pulse input signal is shown in (a). Figure (b) shows the signal after being filtered with a single pole recursive low-pass filter, passing from *left-to-right*. In (c), the signal has been processed in the same manner, except with the filter moving *right-to-left*. Figure (d) shows the signal after being filtered both *left-to-right* and then *right-to-left*. Any recursive filter can be made zero phase by using this technique.

*Daugiau - https://www.dspguide.com/ch19/4.htm*

Matlab aplinkoje „zero-phase“ filtravimui naudojama *filtfilt* funkcija, kuri atlieka signalo filtravimą du kartus – vieną kartą pirmyn, kitą kartą atgaline tvarka.

**Advantages of using *filtfilt*:**

*Zero-phase distortion*: The main advantage of using *filtfilt* over *filter* is that *filtfilt* applies the filter twice, once forward and once in reverse, thereby cancelling out any phase distortion introduced by the filter. This is particularly useful in applications where maintaining the phase relationship between frequency components in a signal is critical, such as in biomedical signal processing.

*Increased filter order*: The forward-backward approach effectively doubles the order of the filter, which might be beneficial in certain applications.

*Sharper cutoff*: Since the filter is applied twice, the cutoff characteristics of the filter might be sharper as compared to applying the filter just once.

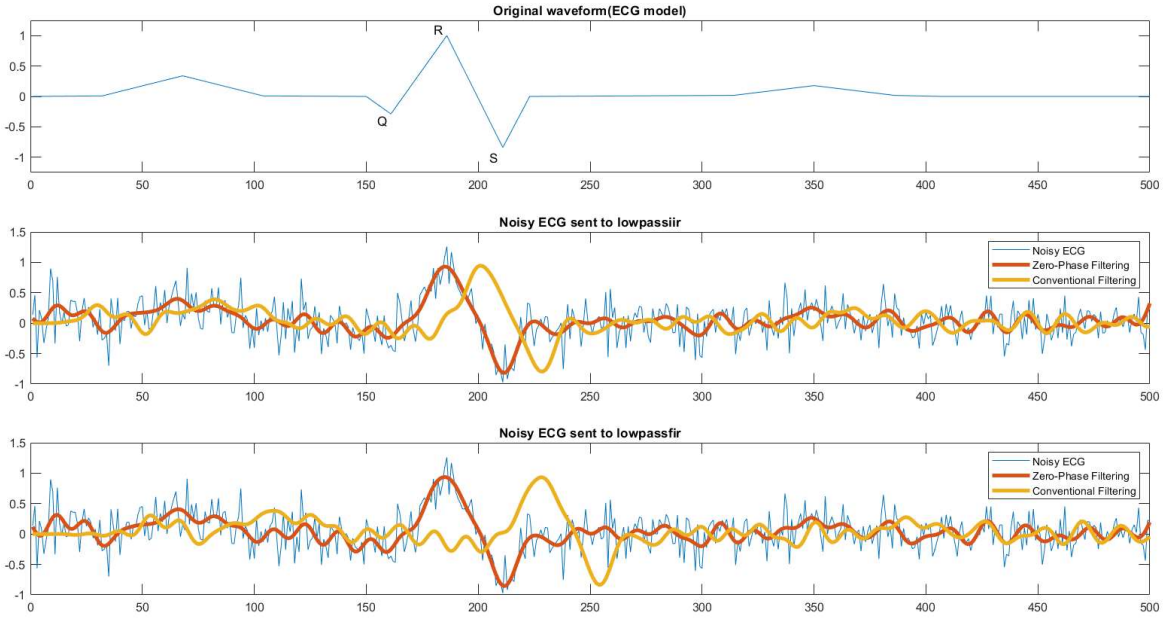
**Disadvantages of using *filtfilt*:**

*Edge effects*: The *filtfilt* function has to deal with the edges of the signal when applying the filter in reverse. It typically uses reflection of the signal to deal with this, but it might cause edge artifacts in the resulting signal, especially for signals with high frequency components.

*Increased computation*: *filtfilt* applies the filter twice, which means it takes roughly twice the computational resources as a regular filter. In large datasets, this could be a significant drawback.

*Altered noise characteristics*: Because of the doubling of the filter order, the noise characteristics of the filtered signal may change. This may or may not be desirable depending on the specific application.

**Example:**




From the plots, you can see the difference in phase distortion and signal smoothing between the *filter* and *filtfilt* functions.

As for when it should or shouldn't be used, it really depends on the application. *filtfilt* is especially useful in applications where the phase of the signal is important, such as in medical and audio signal processing. It might not be the best choice in applications where computational resources are limited or where the noise characteristics of the signal need to be preserved.

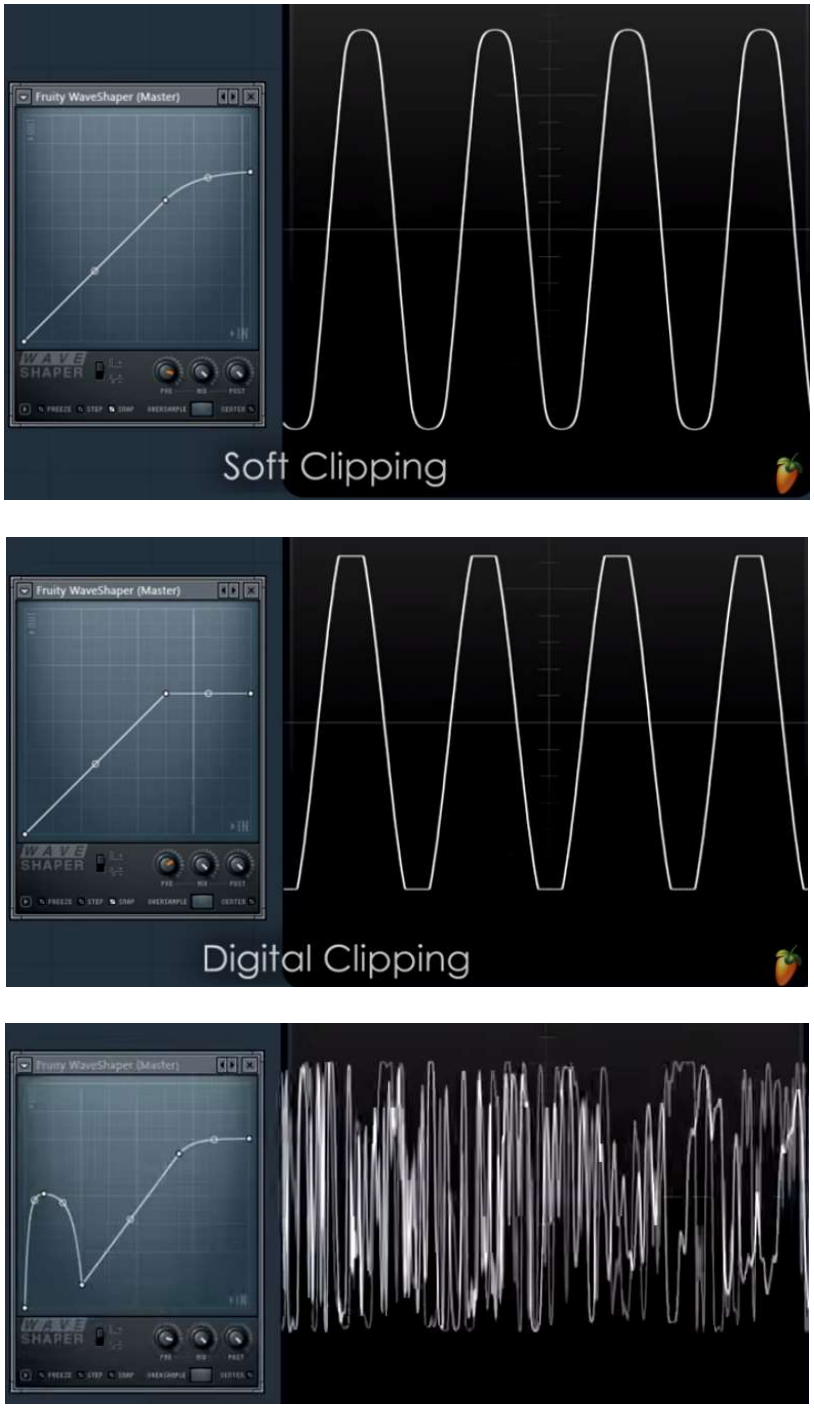
*Šaltinis: https://dsp.stackexchange.com/questions/9467/what-is-the-advantage-of-matlabs-filtfilt/9468#9468*

# Audio signalo efektai

## Iškraipymo (Distortion) efektas

David Guetta – Distortion https://www.youtube.com/watch?v=fQ1b2Ia1x7o

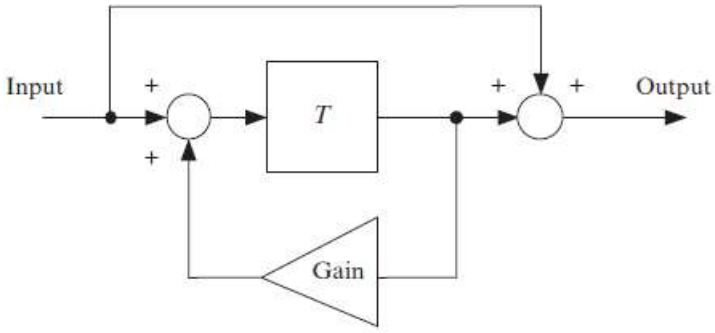
FL Studio Guru | Fruity Waveshaper and Distortion
https://www.youtube.com/watch?v=1L9djVLaUSU




*"Heavy crossover distortion"*

```
//Distortion
int16_t Distortion(int16_t InSample)
{
            float_t OutSample = 0;

            uint16_t high_level = 10000;
            float_t high_gain = 0.5;

            uint16_t low_level1 = 500;
            uint16_t low_level2 = 1500;
            float_t low_gain = 1.0;

            if(abs(InSample) > high_level) OutSample = (float_t)InSample * high_gain;
            else
               if((abs(InSample) > low_level1) && (abs(InSample) < low_level2)) OutSample
= (float_t)InSample * low_gain;
               else
                   OutSample = (float_t)InSample;

            return (int16_t)OutSample;
}
```

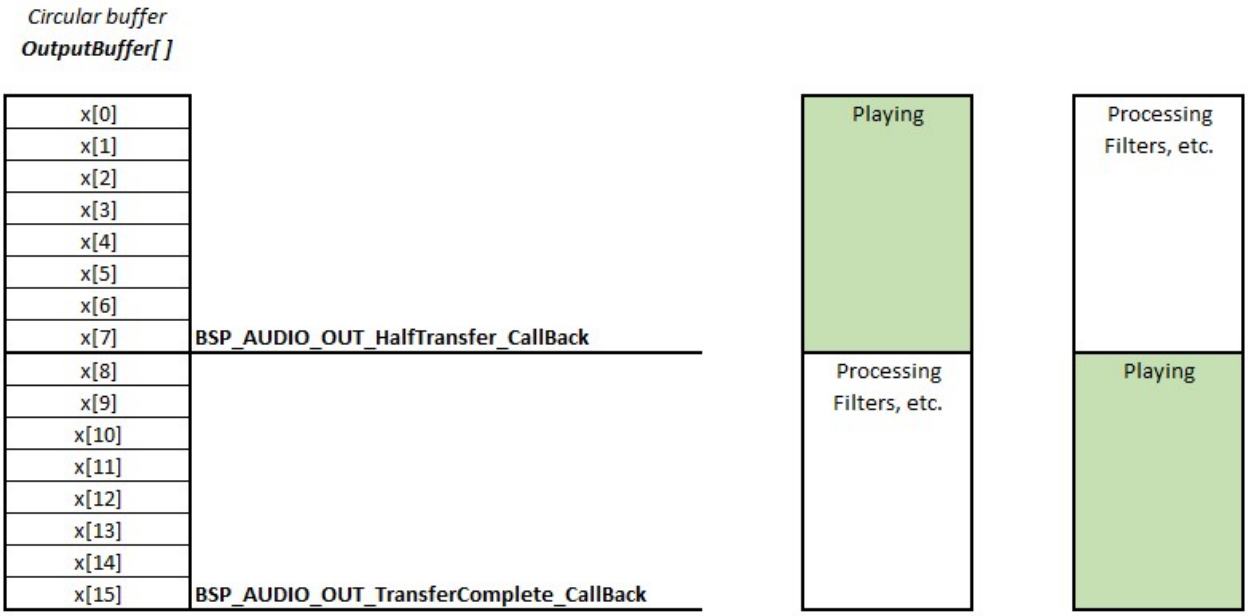
**Aido (Echo) efektas**




3 pav. "*Fading echo*" efekto struktūrinė schema

```
//Echo
int16_t Echo(int16_t InSample)
{
            float_t OutSample = 0;

            OutSample = echo_buffer[echo_ptr];

            echo_buffer[echo_ptr] = (float_t)InSample + OutSample * ECHO_GAIN;
            echo_ptr = (echo_ptr + 1) % ECHO_BUFF_SIZE;

            return (int16_t)OutSample;
}
```

Audio signalo apdorojimas ir atkūrimas realiu laiku realizuotas naudojant žiedinį buferį (*Circular buffer)*.




Žiedinio buferio atveju signalas atkuriamas be perstojo – pasibaigus buferiui reikšmės vėl pradedamos skaityti nuo buferio pradžios.

- Pasiekus buferio vidurį, iškviečiama *HalfTransfer_CallBack* – pertrauktis. Kadangi toliau yra skaitomos 2-ios buferio dalies reikšmės, tuo metu galime manipuliuoti 1-ąja buferio dalimi – perrašyti naujomis reikšmėmis, filtruoti, ir t.t.
- Pasiekus buferio pabaigą, iškviečiama *TransferComplete_CallBack* – pertrauktis. Kadangi toliau skaitomos reikšmės nuo buferio pradžios (iš 1-os dalies), tuo metu galime manipuliuoti 2-ąja buferio dalimi.

Matlab skriptas audio signalui konvertuoti iš .wav failo.
Audio failas parengtas *Audacity* programa – iškirpta 4 sek. trukmės audio kūrinio ištrauka, išsaugota 16 skilčių PCM formatu.

```
%read wav
%[wave,fs]=audioread('guitar_short_16_PCM.wav');
[wave,fs]=audioread('guitar_short_16_PCM_4sec.wav');
t=0:1/fs:(length(wave)-1)/fs;
plot(t,wave);

% ---------------------------------------------
% Save to faile
fid=fopen('audio_data2.txt','w');

% koeficientu kiekis
fprintf(fid, '/* PCM coded audio data */  \r\n', [length(wave)]');
fprintf(fid, 'uint32_t audio_data_size = %d;  \r\n', [length(wave)]');

% saugomi koeficientai + kablelis
fprintf(fid, ['const int16_t audio_data[] = { \r\n']);
    for j=1:length(wave)-1
       %stereo
       %fprintf(fid, '%f,', [wave(j,1)]');
       %fprintf(fid, '%f,', [wave(j,2)]');
       %mono
       %fprintf(fid, '%f,', [wave(j)]');
       fprintf(fid, '%5.0f,', [wave(j)]*30000');
    end

% paskutinis koeficientas be kablelio
%stereo
%fprintf(fid, '%f', [wave(length(wave),1)]');
%fprintf(fid, '%f', [wave(length(wave),2)]');
%mono
%fprintf(fid, '%f', [wave(length(wave))]');
fprintf(fid, '%.0f', [wave(length(wave))]*30000');
% pabaiga + failo uzdarymas
fprintf(fid, [' \r\n };']);
fclose(fid);
```

## Praktinė dalis

- Išanalizuoti pateiktą programą su pasirinkta muzikinio kūrinio ištrauka ir pavyzdiniais filtrais.
- Suprogramuoti papildomai pasirinktą audio efektą (ir naują audio kūrinį)