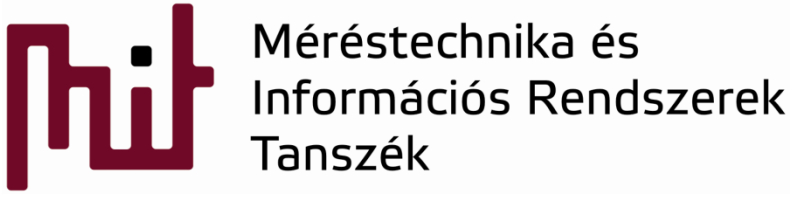
Embedded and ambient systems 2023.12.05.

Implementation of filtering operations



FIR filters

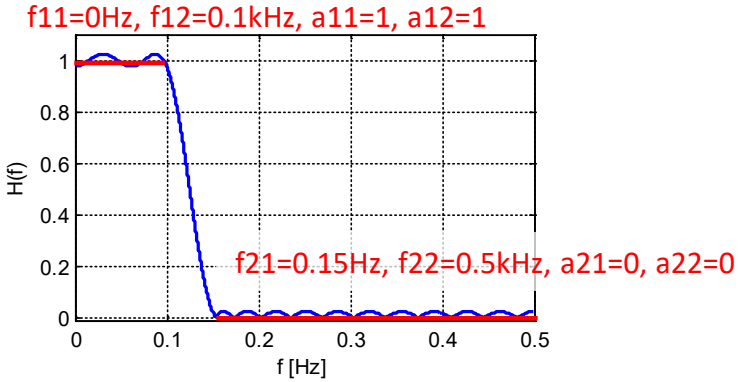
- FIR: Finite Impulse Response (this is a digital filer type)
  - Finite impulse response: contains a finite number of samples
    - Or at least can be truncated to a finite number if the samples “at the end” can be neglected
  - During filter design the impulse response is calculated (w_i), then the convolution sum is evaluated:

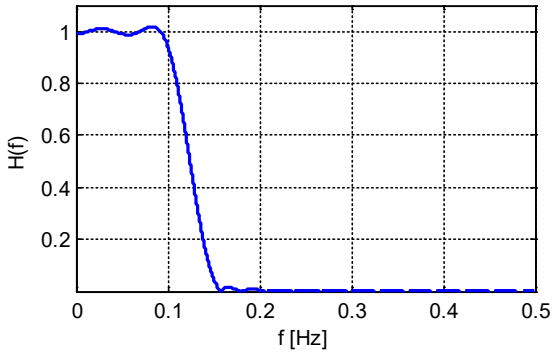
$$y_n = \sum_{i=0}^{N-1} w_i x_{n-i}$$

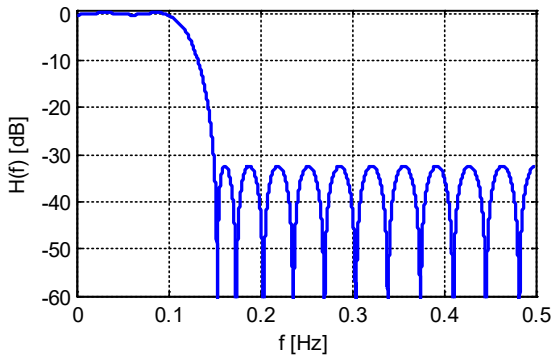
  - In practice: N*10 < number of samples < N*100
    - Exclusions may exist, e.g., acoustic impulse response of a concert hall: N>10000
    - Convolution with some 100 tap → large computation capacity is required☹
  - Feedback is not applied
    - Remains stable regardless the number representation applied ☺
    - Characteristic depends on the number representation applied
  - Linear-phase filter is possible to be designed: delay is frequency independent
    - Signal components with different frequency do not suffer to relative delay: signal shape is unchanged
  - Design methods: MATLAB, Python, octave (MATLAB clone)

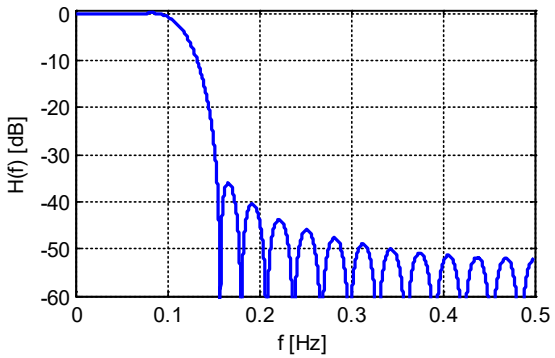
FIR filters: filter types

- Design methods: MATLAB functions , or Python, octave
- Remez algorithm: even fluctuation (ripple) around the specified region
  - w = firpm(N, [f11 f12, f21 f22], [a11 a12, a21 a22]);
- LS algorithm: fits to a characteristic by least squares approach
  - w = firls(N, [f11 f12, f21 f22], [a11 a12, a21 a22]);







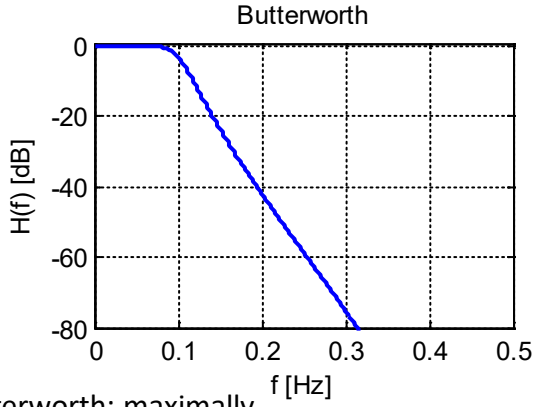


IIR filters

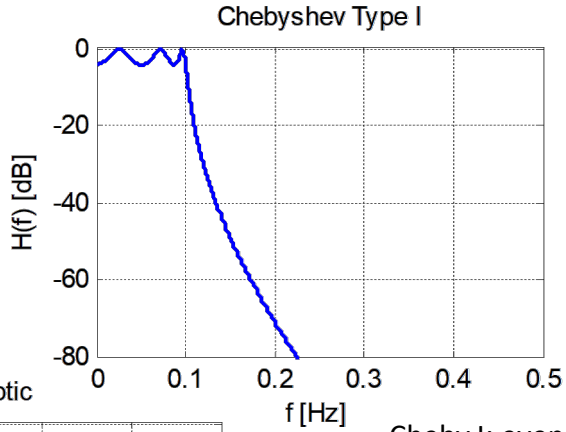
- IIR: Infinite Impulse Response (type of filter)
  - The impulse response is infinitely long
    - In practice after a while the samples reach such small values that can be neglected
  - Can be defined by finite number of parameters
    - In practice: $1 \leq$ tap number < 10
    - Small tap number → small computation capacity is enough☺
  - Due to the feedback applied it is sensitive to the number representation applied ☹
    - Not only the characteristic may change but may become instable due to the rounding error of numbers☹
  - Linear phase is not possible not even theoretically → signal shape is distorted even in the passband
    - There exists some methods that optimize filter design to reach close-to-linear phase characteristics
  - Design methods: MATLAB functions , Python, octave

IIR filters: filter types

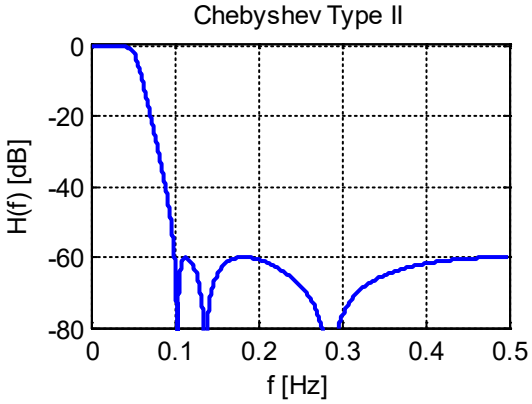
- Butterworth: maximally flat
  - [B,A]=butter(N, fc): N: tap number, fc: corner frequency
- Chebysev {1,2}: even fluctuation (ripple) in the pass- or stopbands (Cheby I.-II.)
  - [B,A]=cheby{1,2}(N,R): N: tap number, R: ripple around the specification
- Elliptic filter: given ripple around the pass-/stop bands. Steep suppression.
- Bessel filter: preserves signal shape, maximally flat



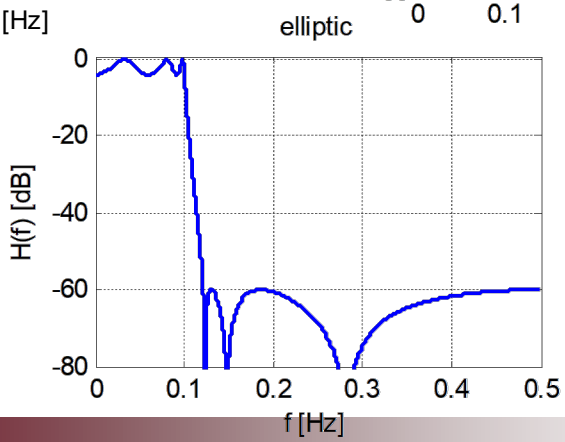
Butterworth: maximally flat characteristics. Week steepness but preserves the signal shape



Cheby I: even fluctuation (ripple) in the passband

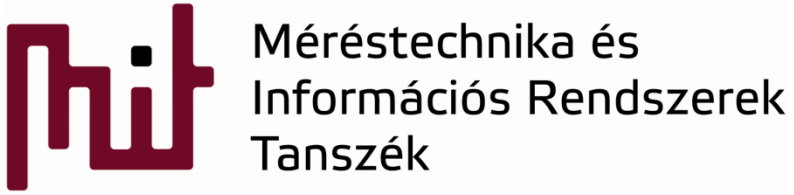


Cheby II: even fluctuation (ripple) in the stopband (minimal suppression can be given, e.g.: 60 dB)



elliptic: ripple in both in the pass- and stop bands. Steep suppression, but the signal is distorted the most.

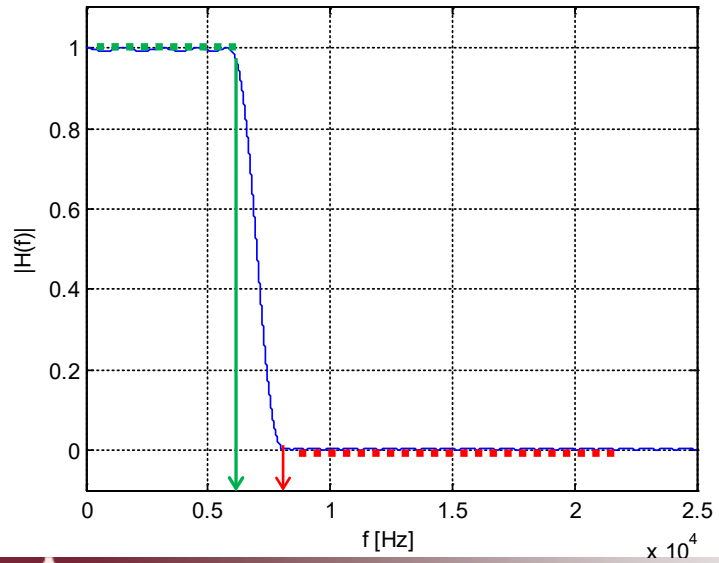
Implementation of FIR filter

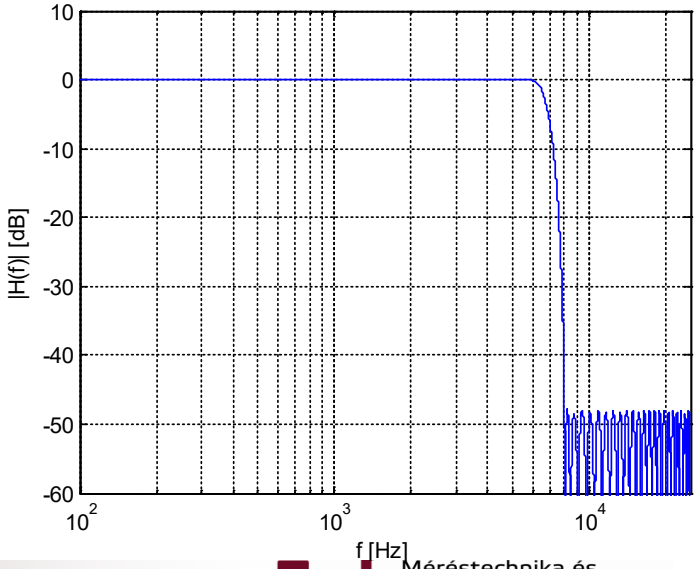


Efficient implementation of FIR filter

- Design of filter coefficients: toolboxes in different programs available (pl. MATLAB, Python, octave…)
  - Example:
    - Design of a 64-tap filter,
    - Sampling frequency: fs=50 kHz
    - Passband: 0…6000Hz,
    - Stopbans: 8000Hz…fs/2

```
fs=50000;  N = 64;
w=firpm(N-1, [0 6000 8000 fs/2]/(fs/2), [1 1 0 0]); % specified for each frequency region
```

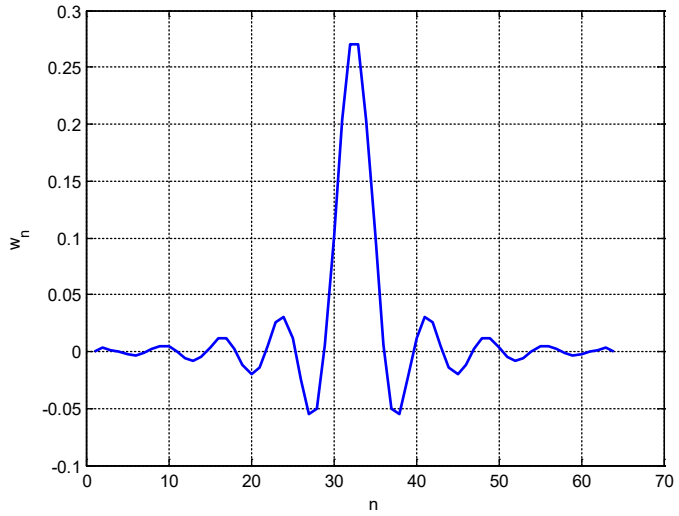




Efficient implementation of FIR filter

- Design of filter coefficients: toolboxes in different programs available (pl. MATLAB, Python, octave…)
  - Example:
    - Design of a 64-tap filter,
    - Sampling frequency: fs=50 kHz
    - Passband: 0…6000Hz,
    - Stopbans: 8000Hz…fs/2

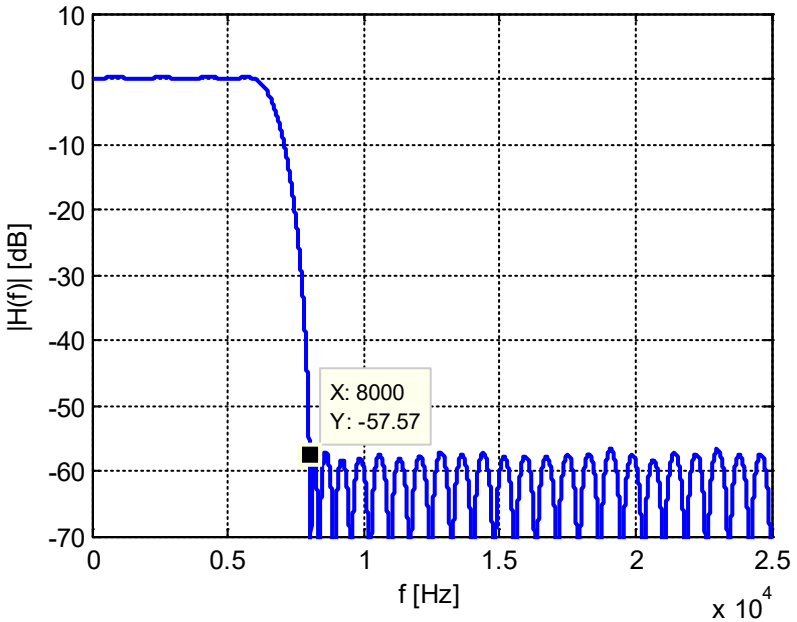
```
fs=50000;  N = 64;
w=firpm(N-1, [0 6000 8000 fs/2]/(fs/2), [1 1 0 0]); % specified for each frequency region
```

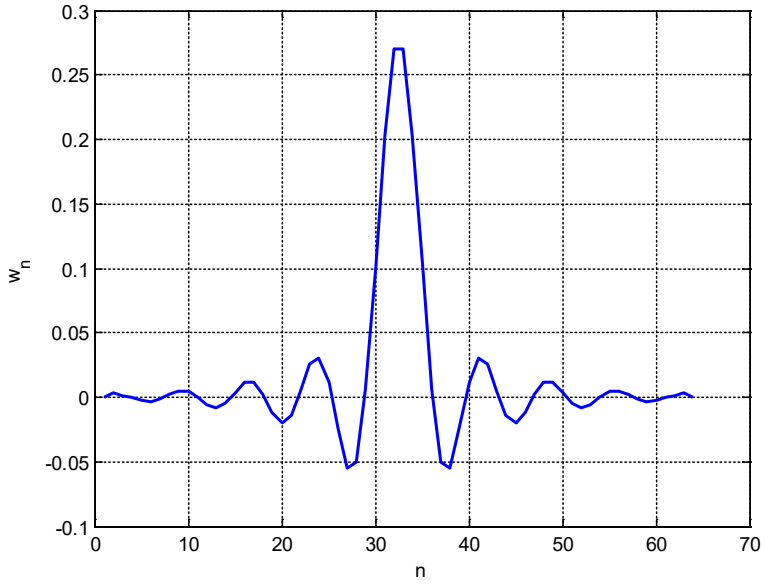


Practical example

- Filter specification:
  - Design of a 64-tap filter,
  - Sampling frequency: fs=50 kHz
  - Passband: 0…6000Hz,
  - Stopbans: 8000Hz…fs/2

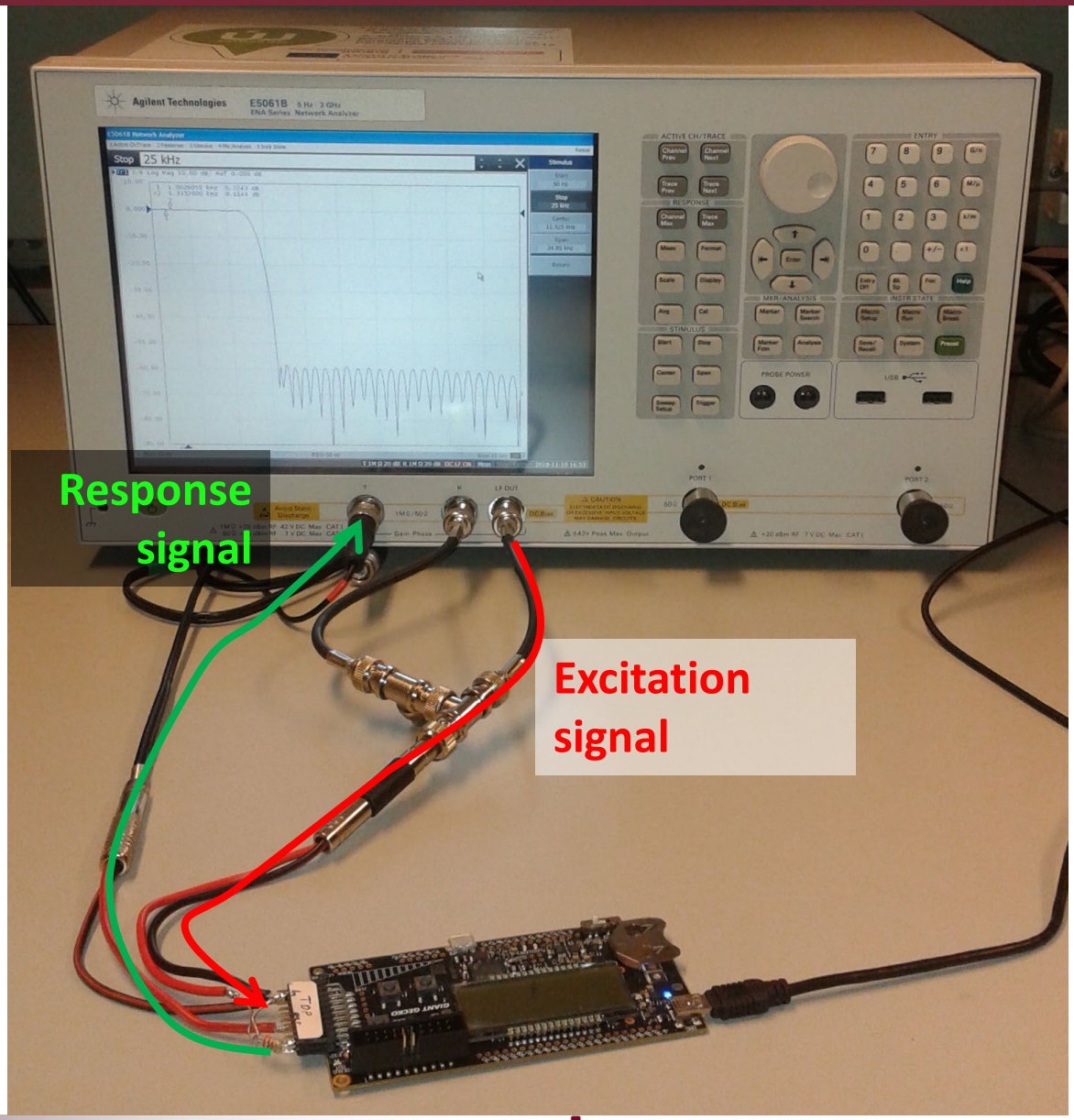
```
N = 64;
w=firpm(N-1, [0 6000 8000 fs/2]/(fs/2), [1 1 0 0]);
```





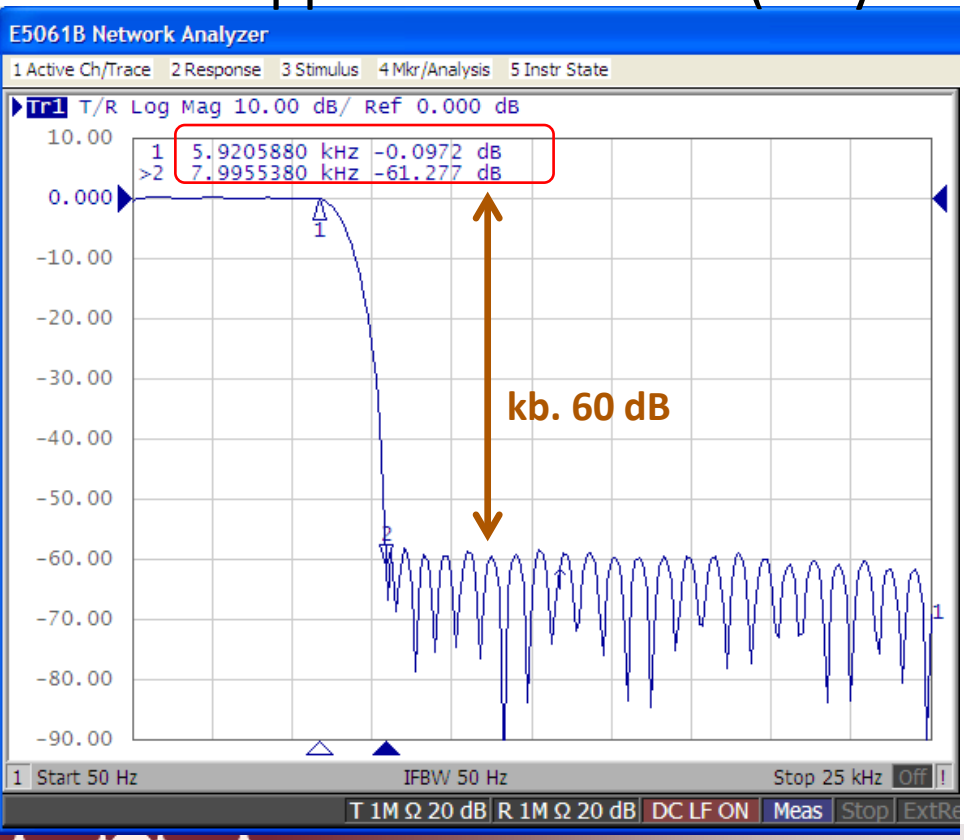
Measurement

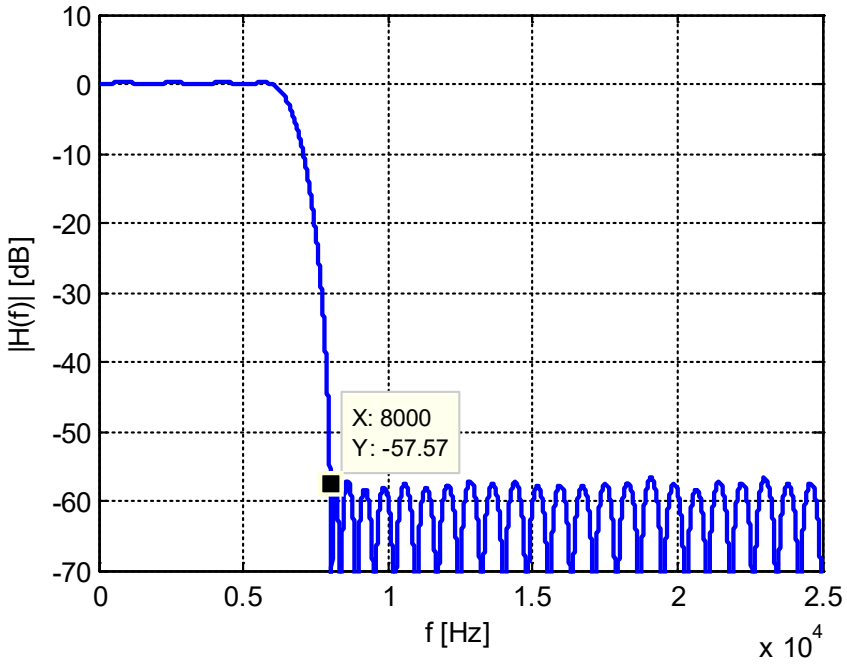
- Measurement by network analyzer
  - Designed specially for the measurement of transfer functions
  - Using a constant amplitude signal with stepped frequency as the excitation to the network of interest, the instrument plots the response of the network (i.e. the transfer function)



Measurement result

- Comparison the implemented and measured filter with the designed filter
  - Stopband starts at: 8 kHz (see markers)
  - Suppression: ~60 dB (very close to the expected value)

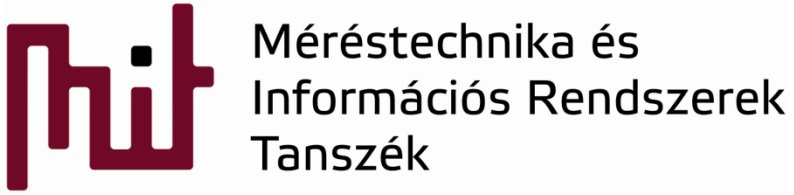




Steps of filtering process

- Filtering operation (convolution): $y_n = \sum_{i=0}^{N-1} w_i x_{n-i}$
- w_i: filter coefficients (values the impulse response)
- x_{n-i}: samples of the signal to be filtered (last N is needed in the n-th *"time"* (better to say *sample*) instant)
- Required operations in every sampling interval:
  - Storing the samples
  - Convolution
    - Running a loop of size N
      - Multiplication of w_i and x_{n-i} numbers
      - Accumulation (addition to the previous) of results of multiplications

Implementation of IIR filters



Structure of IIR filters

- Transfer function of IIR filters:

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M}}{1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}}$$

- From this

$$Y(z)(1 + a_1 z^{-1} + a_2 z^{-2} + \ldots + a_N z^{-N}) = X(z)(b_0 + b_1 z^{-1} + b_2 z^{-2} + \ldots + b_M z^{-M})$$

- In time range:

$$y_n + a_1 y_{n-1} + \ldots + a_N y_{n-N} = b_0 x_n + b_1 x_{n-1} + \ldots + b_M x_{n-M}$$

$$y_n = b_0 x_n + b_1 x_{n-1} + \ldots + b_M x_{n-M} - a_1 y_{n-1} - \ldots - a_N y_{n-N}$$

$$y_n = \sum_{i=0}^{M} b_i x_{n-i} - \sum_{j=1}^{N} a_j y_{n-j}$$

Implementation of IIR filters

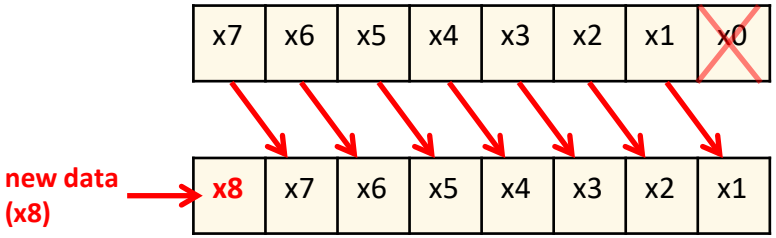
- In its structure, it is essentially similar to an FIR filter: the current output data is given as a sum of products
  - Difference: not only the input data x_n, but also the output data y_n is also used
- The same speed-up techniques as for FIR filters can be used
- Since the degree is usually low (usually N ≤ 10), data storage is less critical

Simple implementation

- Tasks:
  - Storage of incoming data
  - Incoming and old outgoing data convolution
  - Storage of current output data
- In the sample code, we do not use circular due to the small amount of data, instead the data is continuously shifted in a register
- x_IIR: containing input data buffer, y_IIR: output data, B_IIR and A_IIR: numerator and denominator of coefficients (filter coefficients).

```
// data shift
for(ii=N_IIR-1; ii>0; ii--){
x_IIR[ii] =x_IIR[ii-1];
y_IIR[ii] =y_IIR[ii-1];
}
x_IIR[0] = data_in; //first data storing

//filtering operation
out=x_IIR[0]*B_IIR[0];
for(ii=1; ii<N_IIR; ii++){
out+=x_IIR[ii]*B_IIR[ii];
out-=y_IIR[ii]*A_IIR[ii];
}
y_IIR[0] =out; //output data storing
```



Biquad implementation

- Problem: IIR filters are very sensitive to coefficient quantization
- General solution: the transfer function is decomposed into a series of quadratic terms (biquads)
- Example:
  - Specifications: 6-tap elliptic, 1kHz corner frequency, 1dB ripple, 60dB suppression, fs=50kHz sampling frequency
  - 16-bit number representation

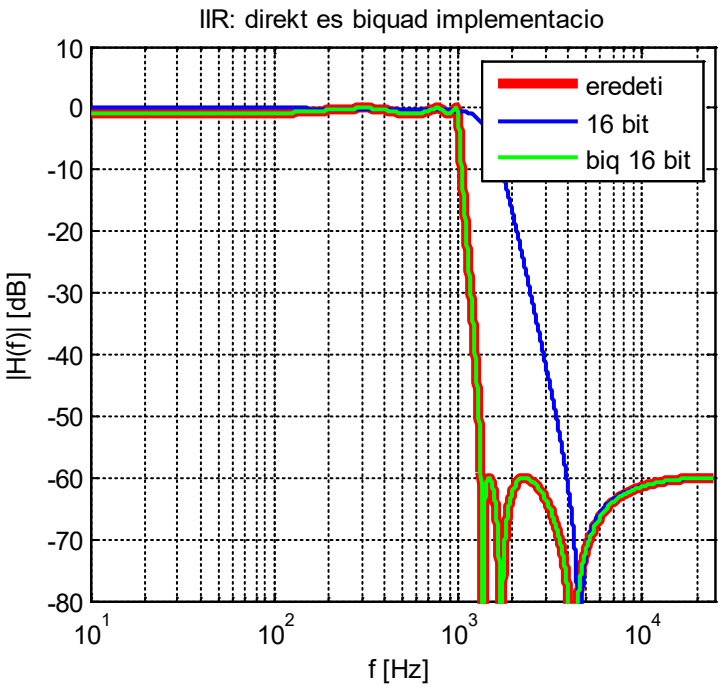
```
[B, A] = ellip(6, 1, 60, 1e3/(fs/2));  %freq(B,A,5000,50e3); : drawing transfer characteristic
[biq, gain] = tf2sos(B, A);
```

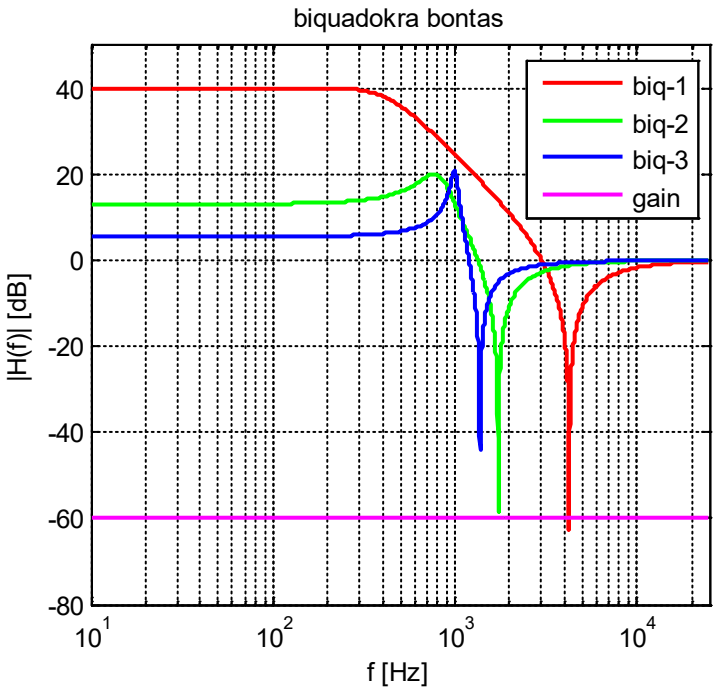
$$H(z)=\frac{1}{973.34}\cdot\frac{1-5.6449z^{-1}+13.6022z^{-2}-17.9144z^{-3}+13.6022z^{-4}-5.6449z^{-5}+z^{-6}}{1-5.8585z^{-1}+14.3276z^{-2}-18.722z^{-3}+13.786z^{-4}-5.4238z^{-5}+0.8907z^{-6}}$$

$$H(z)=\frac{1}{973.34}\cdot\frac{1-1.7217z^{-1}+z^{-2}}{1-1.9312z^{-1}+0.934z^{-2}}\cdot\frac{1-1.9529z^{-1}+z^{-2}}{1-1.9532z^{-1}+0.9635z^{-2}}\cdot\frac{1-1.9703z^{-1}+z^{-2}}{1-1.9741z^{-1}+0.9898z^{-2}}$$

Biquadimplementation

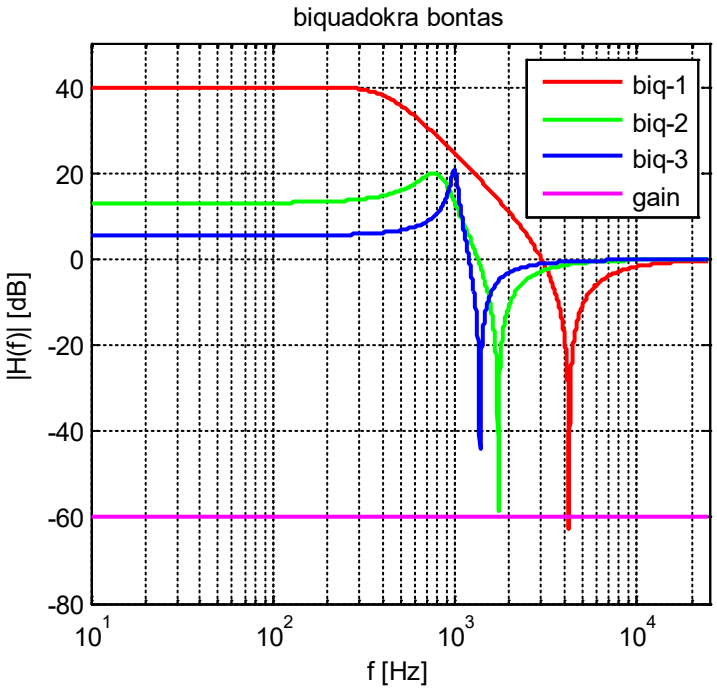
- It can be seen, that the characteristic of the filter of coefficients with rounded to 16 bits is significantly different from ideal (left figure: blue and red characteristics)
- For biquads decomposition even in the case of 16-bit number representation characteristics are essentially the same as ideal (left figure: green and red characteristics)
- Figure on the right: transfer of the individual biquad members individually, and the value of the gain (The gain of the individual members is 40dB, 13dB and 6dB. Therefore, for the resulting unit gain of 0dB, a gain of approx. -60dB is needed: roughly a division of 1000)





Biquad implementation

- Pay attention: for example, in the case of biquad 1 (red characteristic), it has 100 times, i.e. 40dB, amplification.
- Problem: the number of bits is still too large also in the case of biquads (example:giant geckocard)
  - 12 bit input data
  - 16 bit coefficients
  - 100x multiplier: 7 bits
  - 12+16+7=35
  - 35 bits: at least 64-bit number representation
- Although the degree is low, a large bit width is required due to the high dynamics



Biquad implementation

- Generic function to be called with biquas-associated parameters

$$H(z) = \frac{Y(z)}{X(z)} = \frac{b_0 + b_1 z^{-1} + b_2 z^{-2}}{1 + a_1 z^{-1} + a_2 z^{-2}}$$

$$y_n = b_0 x_n + b_1 x_{n-1} + b_2 x_{n-2} - a_1 y_{n-1} - a_2 y_{n-2}$$

```
int64_t
IIR_biq(int64_t*x,int64_t*y,int64_t*B,int64_t*THE,int64_tin){
// input data shift
x[2] =x[1];
x[1] =x[0];
x[0] = in;

// output datas hift
y[2] =y[1];
y[1] =y[0];

//filtering operation
int64_t out= 0;
out=x[0]*B[0] +x[1]*B[1] +x[2]*B[2] -y[1]*THE[1] -y[2]*THE[2];

out= out >> (N_FIX_BIT-1);
y[0] =out;
returnout;
}
```

Program code

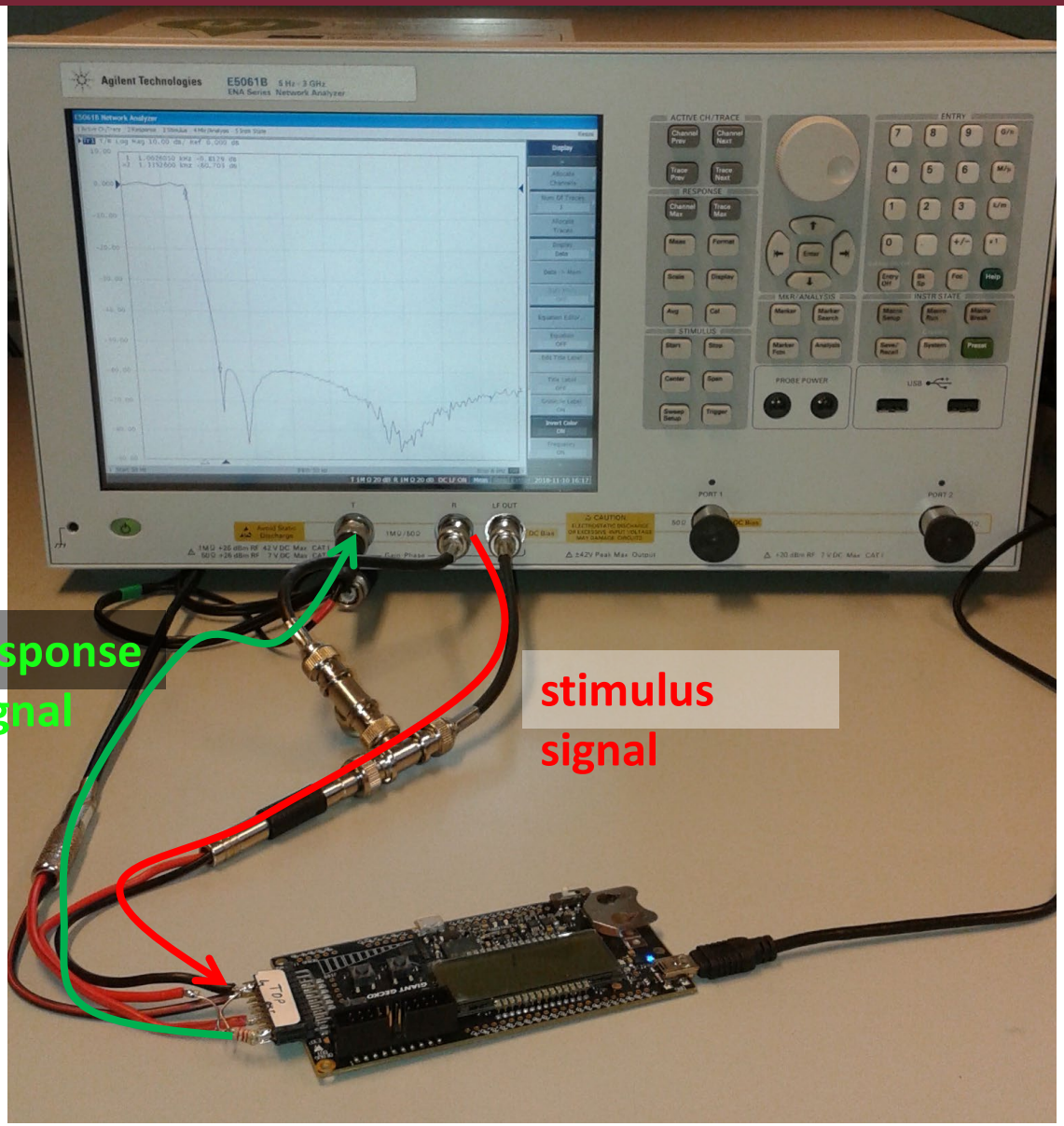
- The implementation of the 6-tap filter by three biquads in series
- Store input and output data in x_biq and y_biq in blocks
  - x_biq[bq][i] means i-th element of the the bq-th biquad
- Store the coefficients in coef_biqin array in the format given by MATLAB
  - coef_biq[bq][0..2]: the bq-th biquad coefficients B
  - coef_biq[bq][3..5]: the bq-th biquad coefficients A

1.0000	-1.7217	1.0000	1.0000	-1.9312	0.9340
1.0000	-1.9529	1.0000	1.0000	-1.9532	0.9635
1.0000	-1.9703	1.0000	1.0000	-1.9741	0.9898

```
int32_tprocess_IIR_biq(int32_t data_in){
int bq;
int64_t out= data_in;
int64_tbq_out_0=0, bq_out_1=0;
bq=0;
bq_out_0=IIR_biq(&x_biq[bq][0], &y_biq[bq][0], &coef_biq[bq][0], &coef_biq[bq][3], data_in);
bq=1;
bq_out_1=IIR_biq(&x_biq[bq][0], &y_biq[bq][0], &coef_biq[bq][0], &coef_biq[bq][3], bq_out_0);
bq=2;
out=IIR_biq(&x_biq[bq][0], &y_biq[bq][0], &coef_biq[bq][0], &coef_biq[bq][3], bq_out_1);
out= out /BIQ_GAIN_REC; //gain
out= out>((1<<12)-1)? ((1<<12)-1) : (out<0? 0 :out); //saturation
return out;
}
```

Measurement

- Measurement using a network analyzer
  - A device specially developed for measuring the transfer function
  - With a constant amplitude signal, the frequency is stepped over a given band and displays the output value (transmission) of the network at the given frequency



IIR filter: measurement result

- Specifications met?:
  - The beginning of the stop band is approx. 1365 Hz
  - About 60 dB suppression: it is in good accordance with the design
- Runtime: 790 CLKs
  - It is comparable to the runtime of the 64-tap FIR filter
  - Fewer operations, but large number of bits are required→ requires a large computing capacity