Embedded and Ambient Systems
2023.11.27.

Moving average, exponential average

Méréstechnika és
Információs Rendszerek
Tanszék

Budapest University of Technology and Economics
Department of Measurement and Information Systems

© BME-MIT

MŰEGYETEM 1782

Moving average

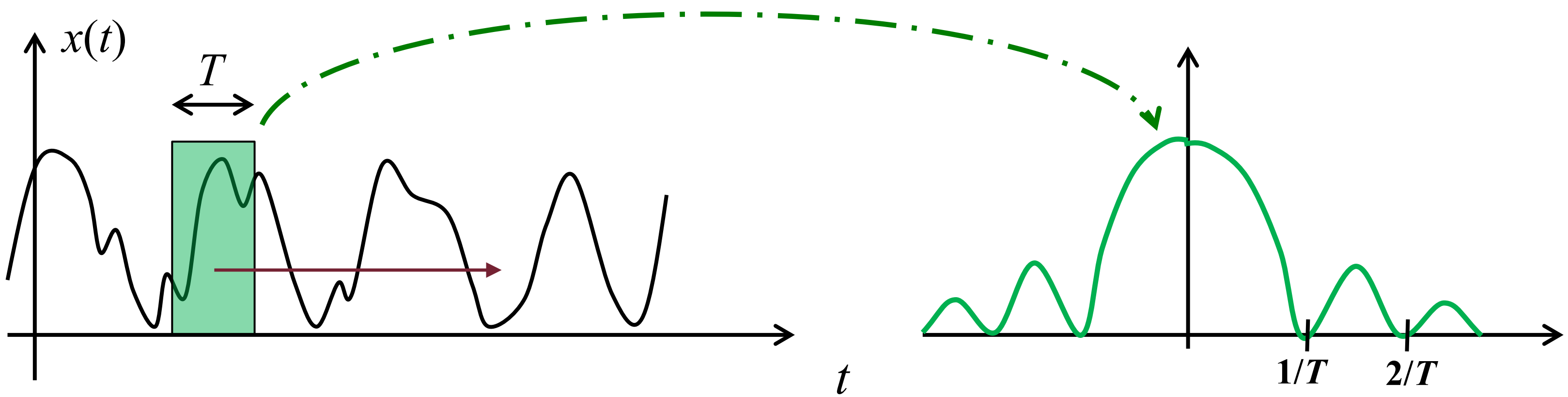
Méréstechnika és Információs Rendszerek Tanszék

© BME-MIT

Continuous-time averaging: transfer func.

- Averaging in continuous time
- Transfer function: sinc(x)=sin(x)/x
  - Zeros of transfer function are found at f=*k*/T where *k* is an integer

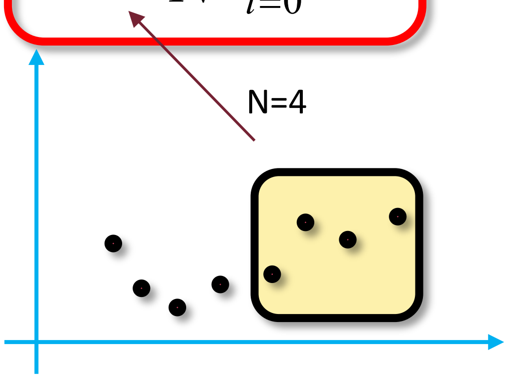
$$y(t) = \int_{t-T/2}^{t+T/2} x(\tau)\,d\tau \quad \dashrightarrow \quad W(s) = T\frac{\sin(\pi f T)}{\pi f T}$$



Averaging

- Averaging is applied not on the whole series of data but only on its windowed part. The window is kept moving by one sample: instantaneous average (expected value)
- Basic signal processing method
- Calculates the average of samples:

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



- Can be considered as digital filtering
  - FIR (finite impulse response) filter
  - Filter coefficients: $w_i = \frac{1}{N}$
  - Convolution: $y_n = \sum_{i=0}^{N-1} w_i x_{n-i} = \sum_{i=0}^{N-1} \frac{1}{N} x_{n-i} = \frac{1}{N}\sum_{i=0}^{N-1} x_{n-i}$

Discrete-time averaging: transfer func.

- Impulse response: $w_i = \dfrac{1}{N}$
- Transfer function of discrete-time averaging

$$W(z) = \sum_{i=0}^{N-1} \frac{1}{N} z^{-i} = \frac{1}{N} \sum_{i=0}^{N-1} z^{-i} = \frac{1}{N} \frac{1-z^{-N}}{1-z^{-1}} = \frac{1}{N} \frac{1-e^{-j\Theta N}}{1-e^{-j\Theta}}$$

$$W(z) = \frac{1}{N} \cdot \frac{e^{-j\frac{\Theta N}{2}}}{e^{-j\frac{\Theta}{2}}} \cdot \frac{e^{j\frac{\Theta N}{2}} - e^{-j\frac{\Theta N}{2}}}{e^{j\frac{\Theta}{2}} - e^{-j\frac{\Theta}{2}}} = \frac{1}{N} e^{-j\Theta\frac{(N-1)}{2}} \frac{\left(e^{j\frac{\Theta N}{2}} - e^{-j\frac{\Theta N}{2}}\right)/2j}{\left(e^{j\frac{\Theta}{2}} - e^{-j\frac{\Theta}{2}}\right)/2j}$$

- Where θ is the dicrete-time angular frequency [0…2π] and $z = e^{j\Theta}$

Discrete-time averaging: transfer func.

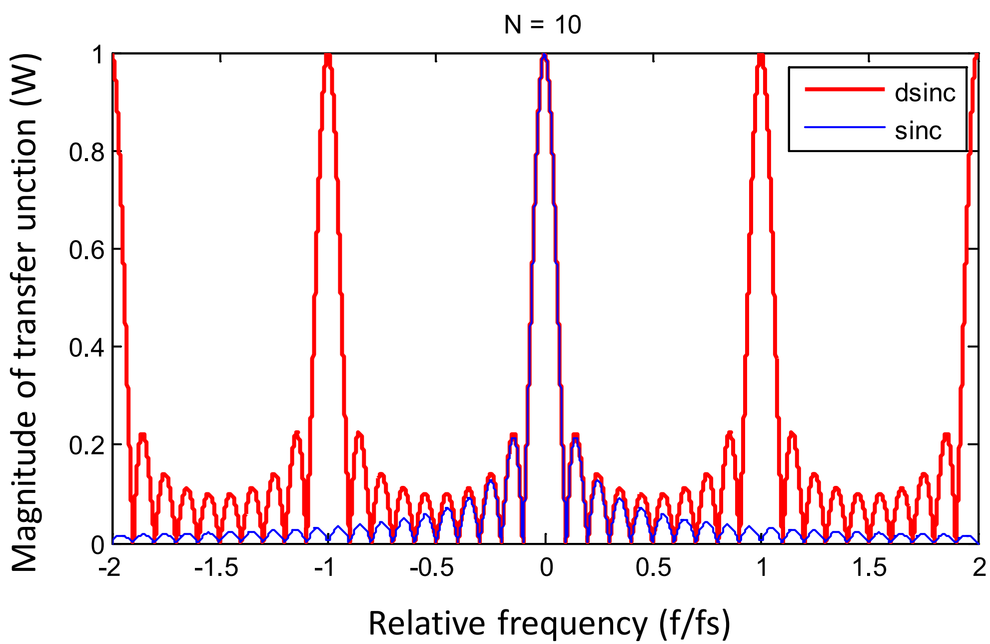
- Transfer function:

$$W(z)=\frac{1}{N}e^{-j\Theta\frac{(N-1)}{2}}\frac{\left(e^{j\frac{\Theta N}{2}}-e^{-j\frac{\Theta N}{2}}\right)/2j}{\left(e^{j\frac{\Theta}{2}}-e^{-j\frac{\Theta}{2}}\right)/2j}=\frac{1}{N}e^{-j\Theta\frac{(N-1)}{2}}\boxed{\frac{\sin\left(\frac{\Theta N}{2}\right)}{\sin\left(\frac{\Theta}{2}\right)}}$$

- This is the discrete-time sinc function
- Phase: ϴ(N–1)/2
  - So the delay is (N–1)/2 samples
- Can be considered as if sinc function was repeated by sampling frequency and being added accordingly

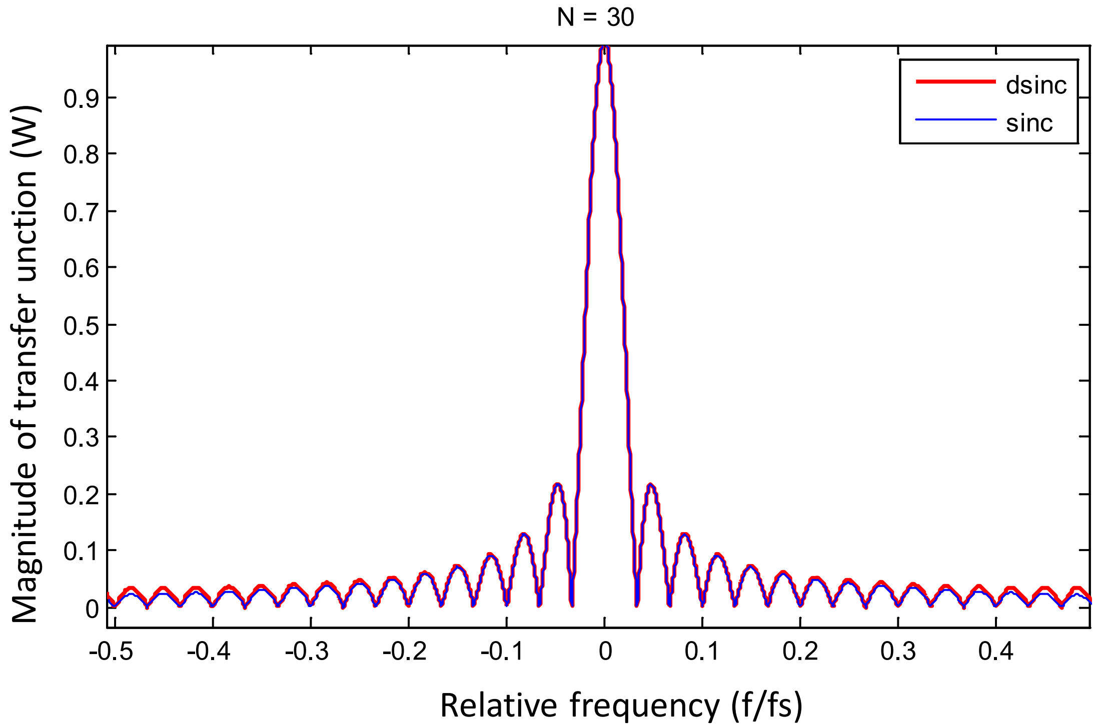
Discrete-time averaging: transfer func.

- Characterisitcs of discrete-time sinc function:
  - Unit value at 0Hz and at integer every multiple of fs
  - Decreases by approximately 1/x envelope
  - Zero places: at every fs/N (e.g. considering fs=10kHz sampling rate and N=20 samples, the zero places are at every integer of 500Hz)
  - If the zero place is required to appear at f0 then N=fs/f0



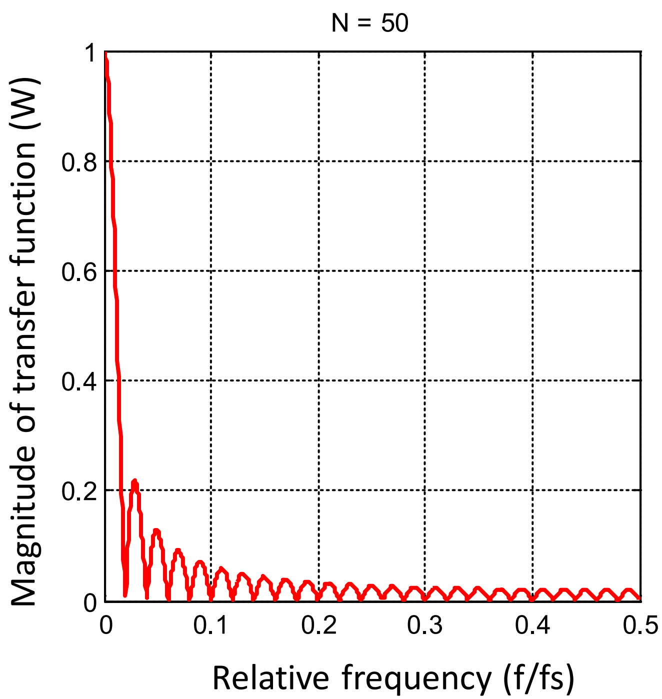
Discrete-time averaging: transfer func.

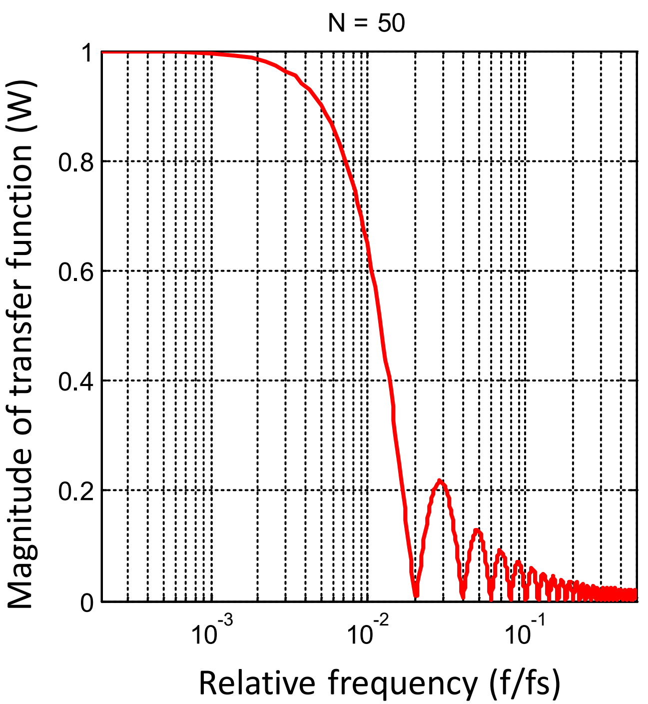
- Approximately in case of N=n*10 samples *sinc* and *dsinc* are very similar in the [-0.5…0.5]·fs frequency region which is our interest



Discrete-time averaging: transfer func.

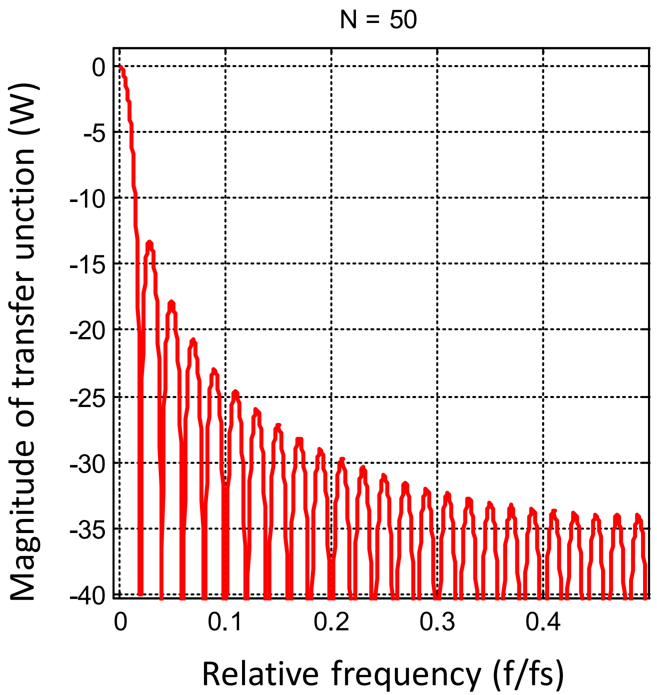
- Disadvantages:
  - In the passband the transfer function is not flat, but increasing continuously, therefore the signal may be distorted
    - Can be used well when the signal is narrowband compared to the noise/disturbance since sharp cutoff cannot be assured
  - In the stop-band the suppression is not flat but approximately reduced by 1/f
    - Can be used well when the signal is narrowband compared to the noise/disturbance since sharp cutoff cannot be assured

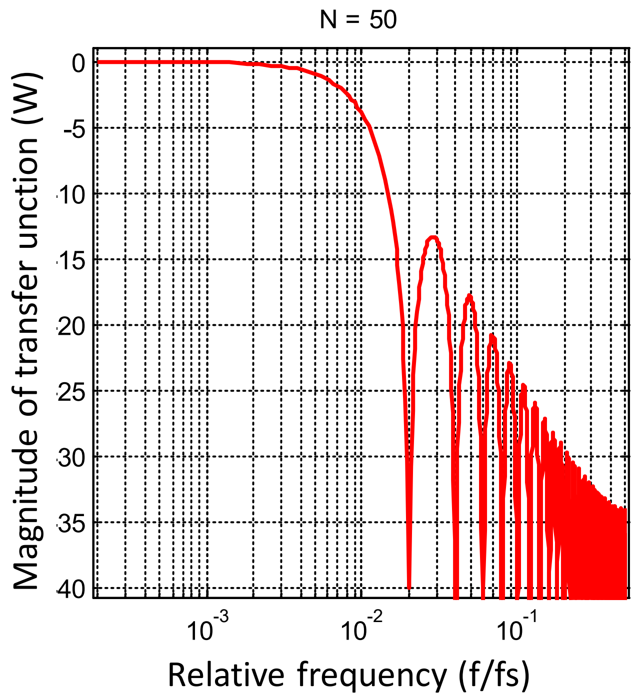




Discrete-time averaging: transfer func.

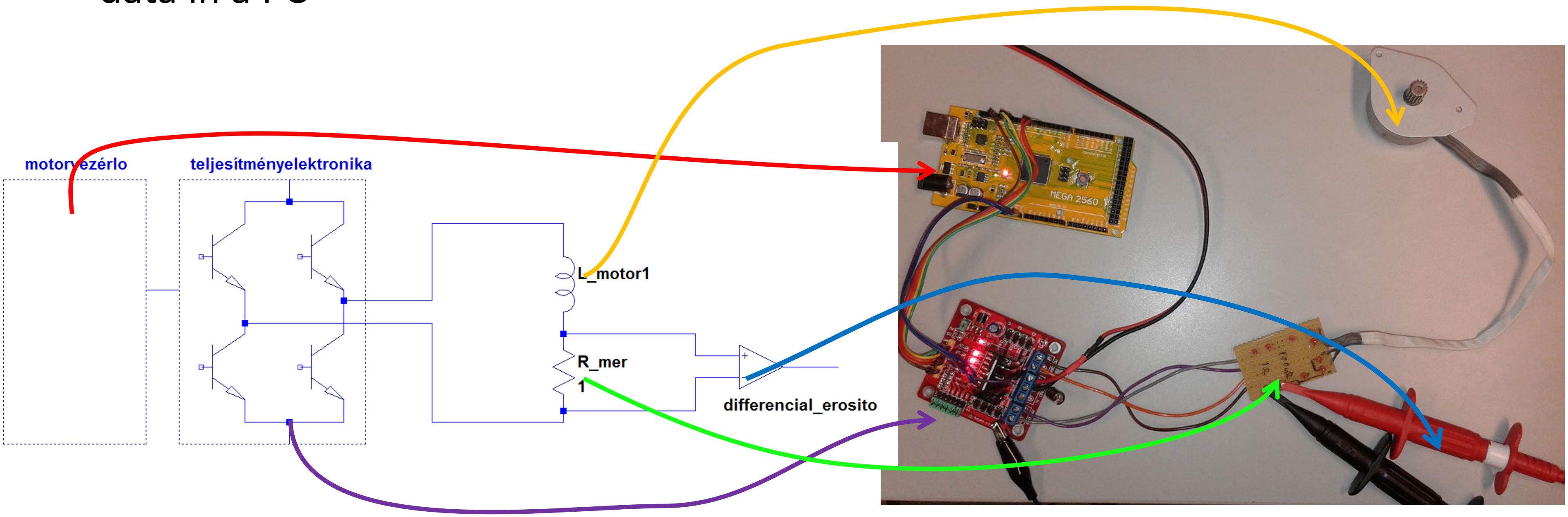
- Disadvantages:
  - In the passband the transfer function is not flat, but increasing continuously, therefore the signal may be distorted
    - Can be used well when the signal is narrowband compared to the noise/disturbance since sharp cutoff cannot be assured
  - In the stop-band the suppression is not flat but approximately reduced by 1/f
    - Can be used well when the signal is narrowband compared to the noise/disturbance since sharp cutoff cannot be assured





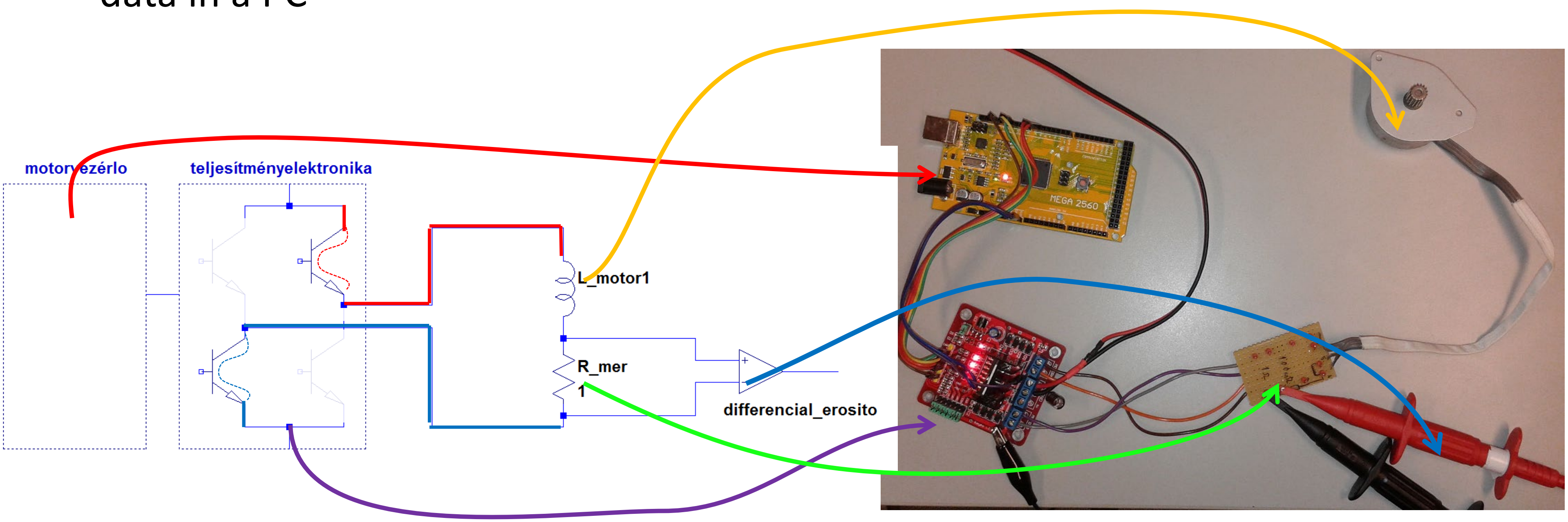
Design example

- Example: measurement of current of a 2-phase stepping motor
- Drive: PWM (pulse width modulation) signal is used to generate periodical current that makes the motor work
- Measurement: voltage on a 1-Ω resistor connected in series -> current is easily calculated
- Measurement in the example: using a special measurement cable integrated with a differential amplifier to measure signal by a digital oscilloscope and store data in a PC



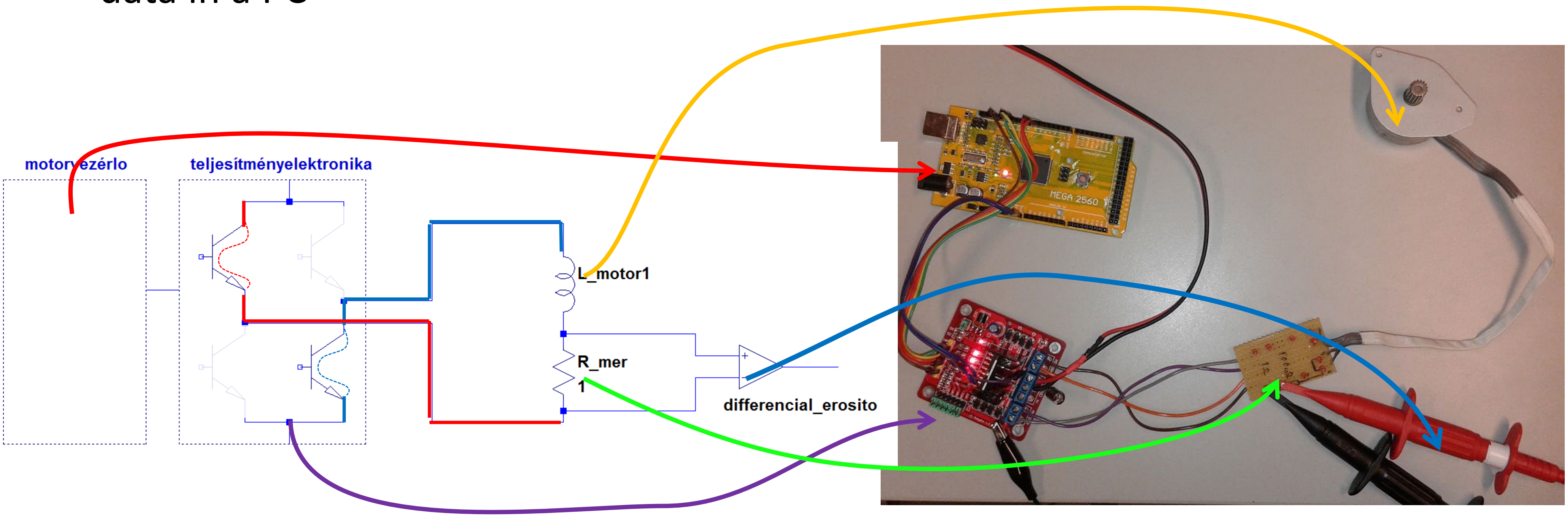
Design example

- Example: measurement of current of a 2-phase stepping motor
- Drive: PWM (pulse width modulation) signal is used to generate periodical current that makes the motor work
- Measurement: voltage on a 1-Ω resistor connected in series -> current is easily calculated
- Measurement in the example: using a special measurement cable integrated with a differential amplifier to measure signal by a digital oscilloscope and store data in a PC



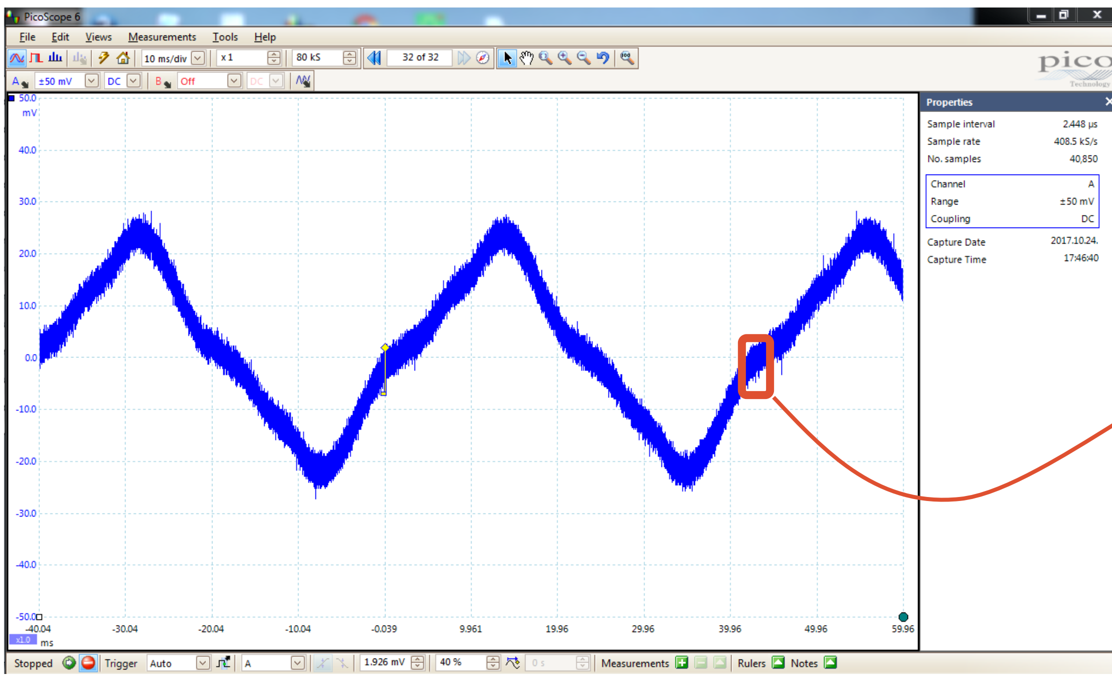
Design example

- Example: measurement of current of a 2-phase stepping motor
- Drive: PWM (pulse width modulation) signal is used to generate periodical current that makes the motor work
- Measurement: voltage on a 1-Ω resistor connected in series -> current is easily calculated
- Measurement in the example: using a special measurement cable integrated with a differential amplifier to measure signal by a digital oscilloscope and store data in a PC



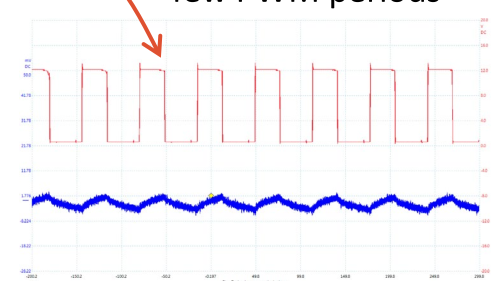
Design example

- Goal: measurement of short-term average value (instantaneous expected value) of current signal
- Disturbances:
  - Noise of differential amplifier
  - Switching transients
  - Current fluctuations due to PWM switching



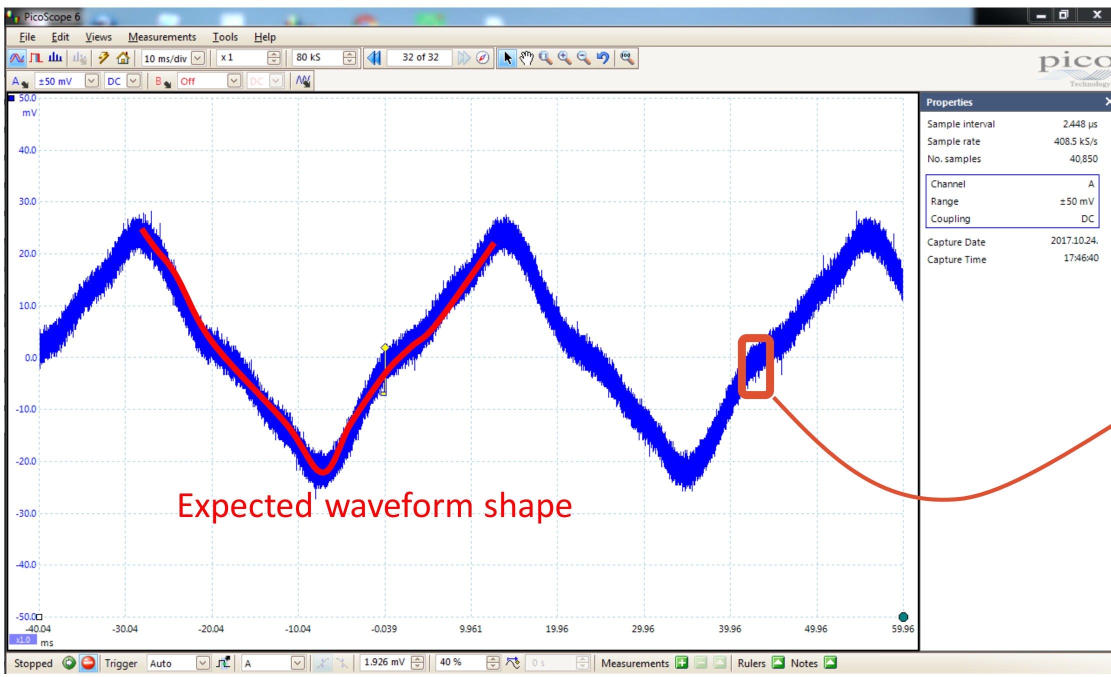
Time function of current signal. Due to the imple implementation the shape is not a sinusoid but a triangle. Frequency is approx. 25 Hz.

Zoomed-in to the current signal for a few PWM periods

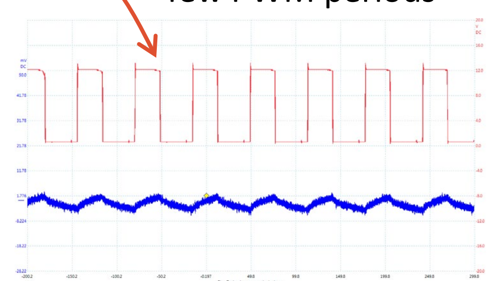


Design example

- Goal: measurement of short-term average value (instantaneous expected value) of current signal
- Disturbances:
  - Noise of differential amplifier
  - Switching transients
  - Current fluctuations due to PWM switching



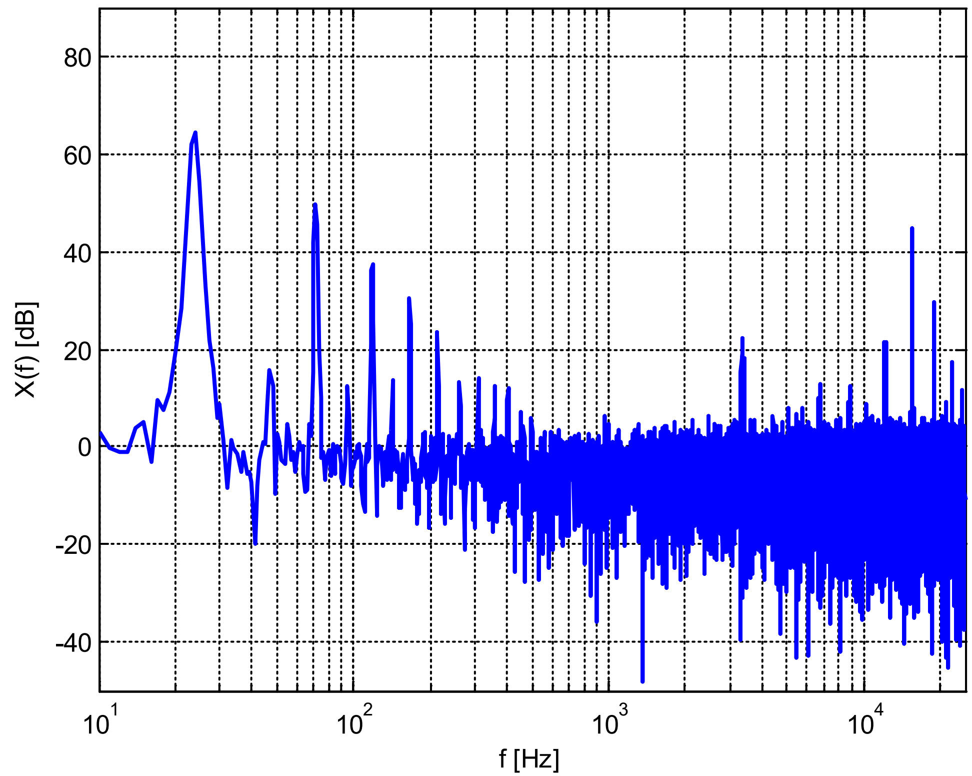
Zoomed-in to the current signal for a few PWM periods



Time function of current signal. Due to the imple implementation the shape is not a sinusoid but a triangle. Frequency is approx. 25 Hz.

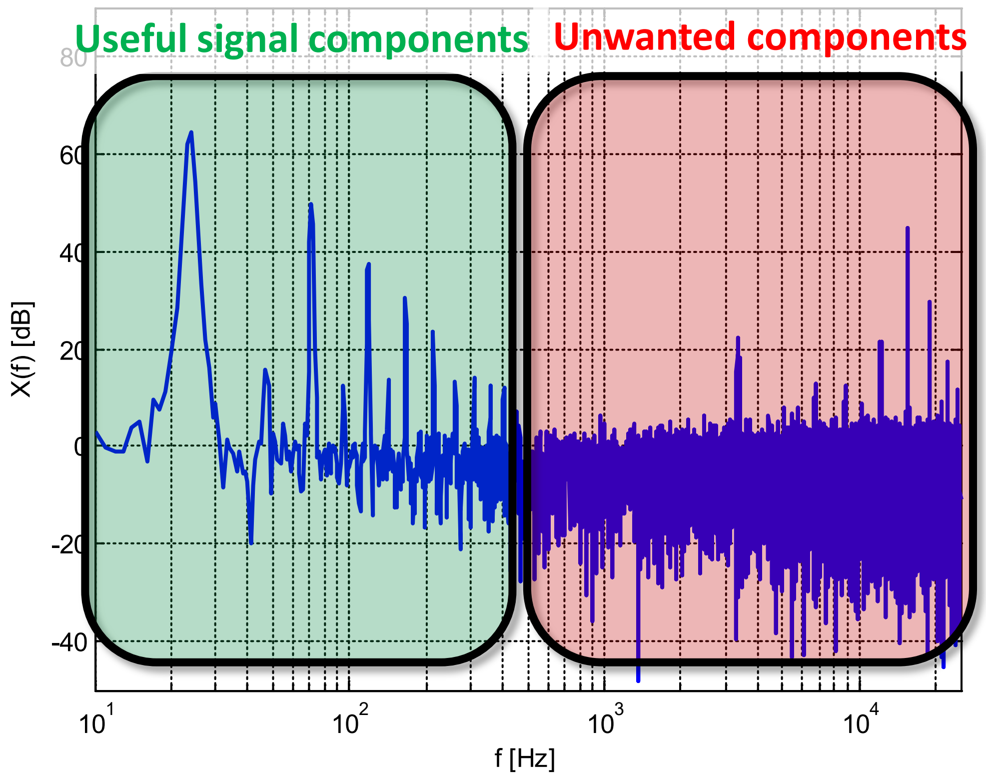
Design: signal analysis

- Current signal is stored to measure its spectrum (i.e. frequency domain representation)
- Results:
  - Useful signal components can be found up to 400-500Hz
    - It is a priori known by the time domain measurements that the useful signal is a periodic one. The spikes appearing at higher frequencies are the harmonics of the periodic signal.
  - Unwanted signal components are found above 500Hz. (Spectrum components at around 20kHz are due to the PWM signal)



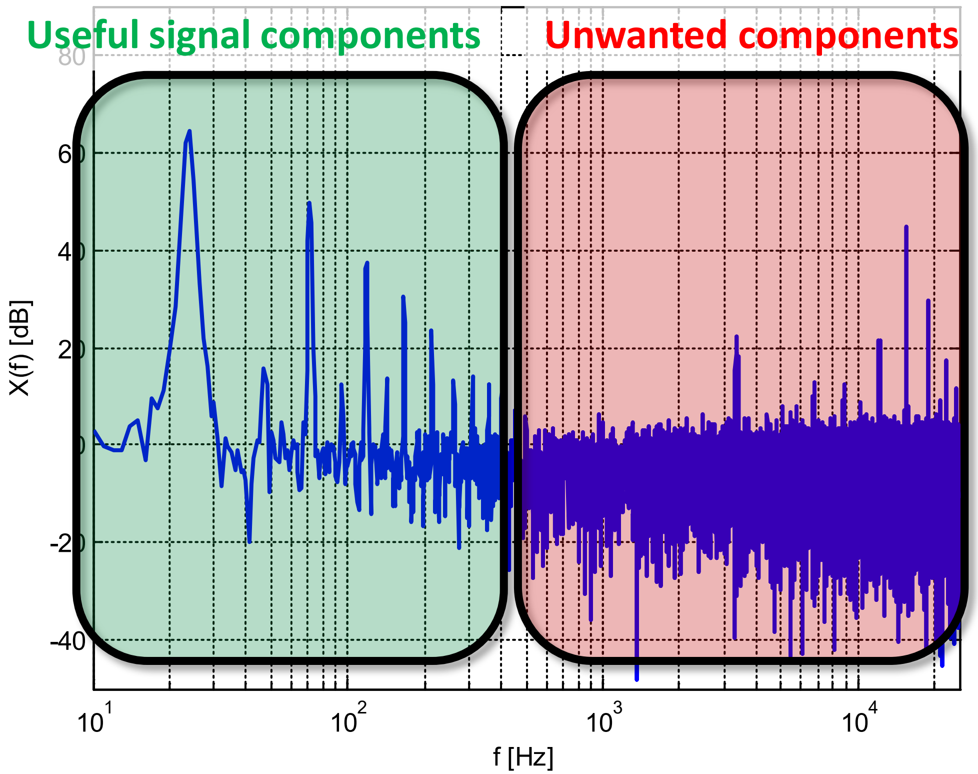
Design: signal analysis

- Current signal is stored to measure its spectrum (i.e. frequency domain representation)
- Results:
  - Useful signal components can be found up to 400-500Hz
    - It is a priori known by the time domain measurements that the useful signal is a periodic one. The spikes appearing at higher frequencies are the harmonics of the periodic signal.
  - Unwanted signal components are found above 500Hz. (Spectrum components at around 20kHz are due to the PWM signal)



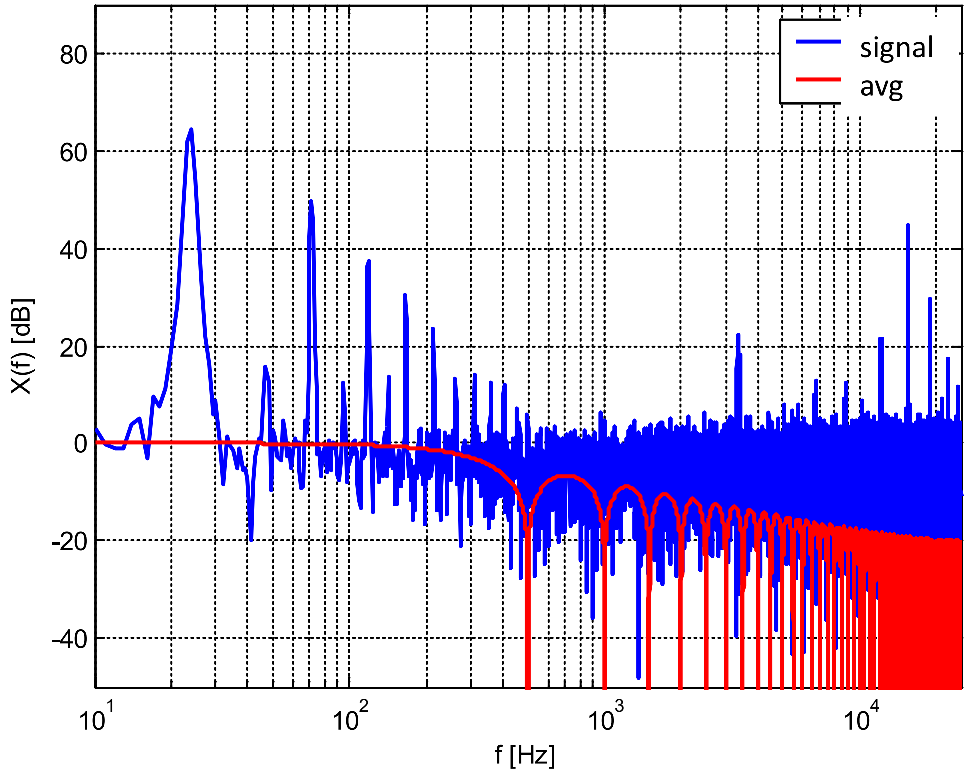
Design: calculation of parameters

- Parameter for averaging: N order
- Let the first zero place be at the end of useful frequency band, i.e., at approx. 500Hz
- Assuming 50kHz sampling frequency: N = 50kHz/500 = 100
  - Remarks: if there exist disturbing periodical components (e.g. signal components due to PWM switching frequency at around 20kHz here), it makes sense to set zero place of transfer function at the frequency where disturbance appears, i.e.: f_unwanted=n*fs/N (N: order, n: one of the zero places)



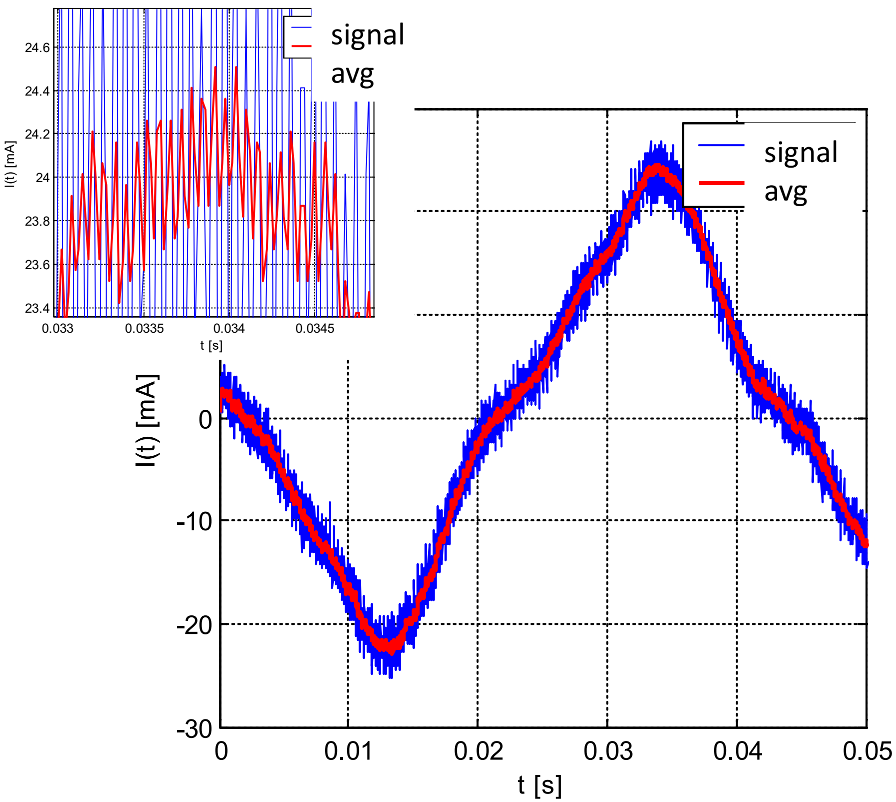
Design: checking in the frequency domain

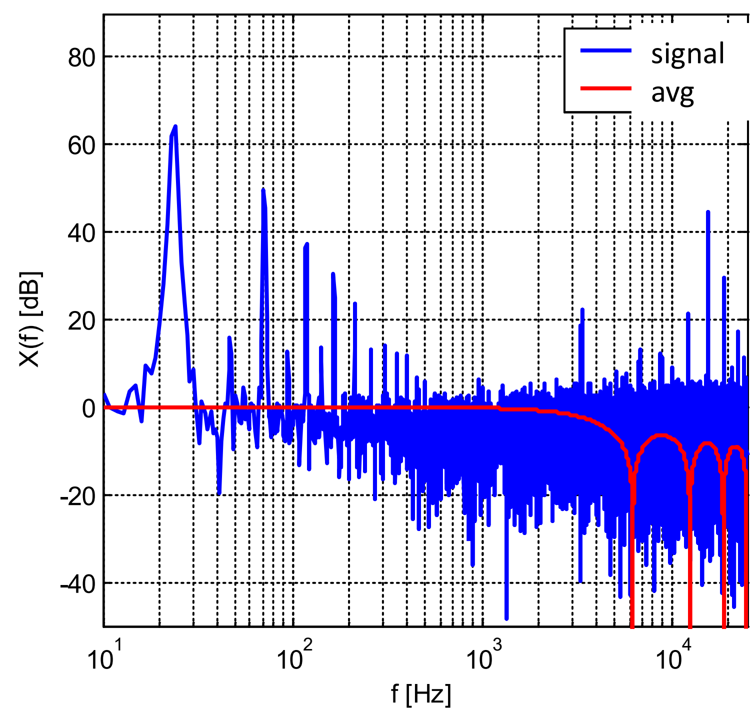
- Checking the transfer function of the designed averaging procedure
- Figure shows signal spectrum (blue) and the transfer function of averaging (red)
- Averaging slightly distorts the useful signal: check the signal in the time domain, to see the distortion whether it is acceptable or not
  - Hint: for efficient implementation let N be the power of 2



Design: simulation in the time domain

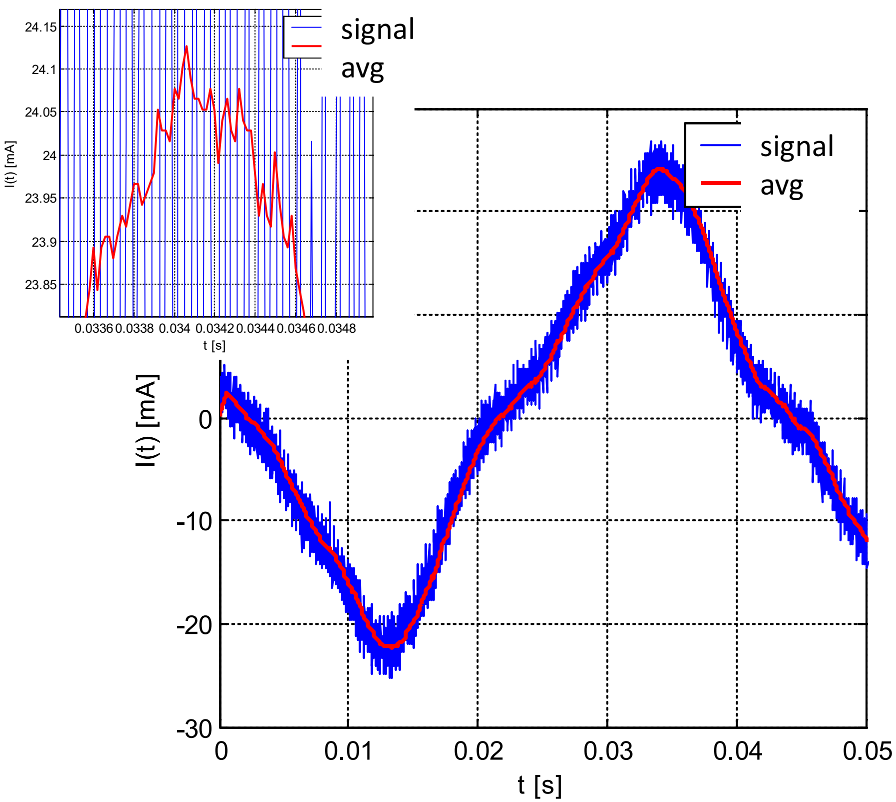
- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- N=8: slight improvement, but the signal is still noisy (approx. 0.6 mA_{pp})

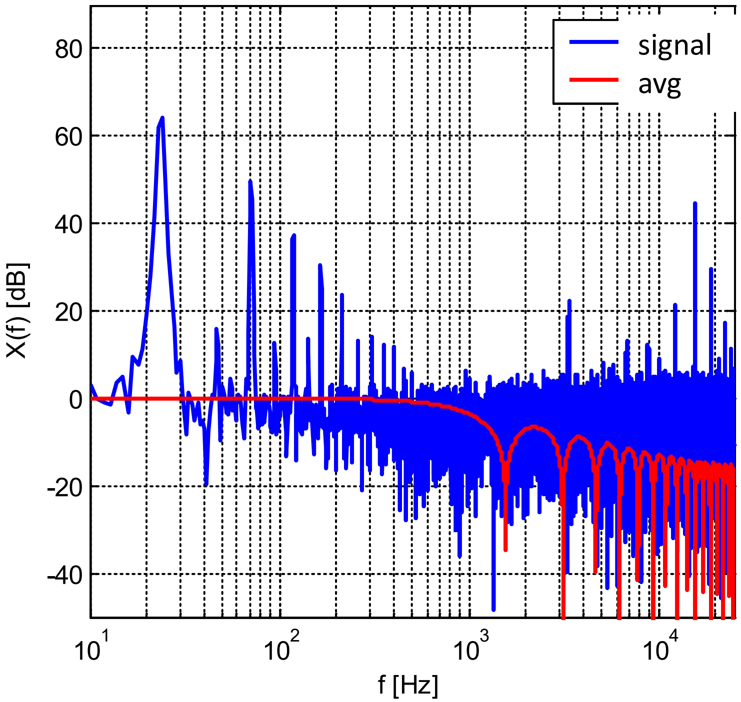




Design: simulation in the time domain

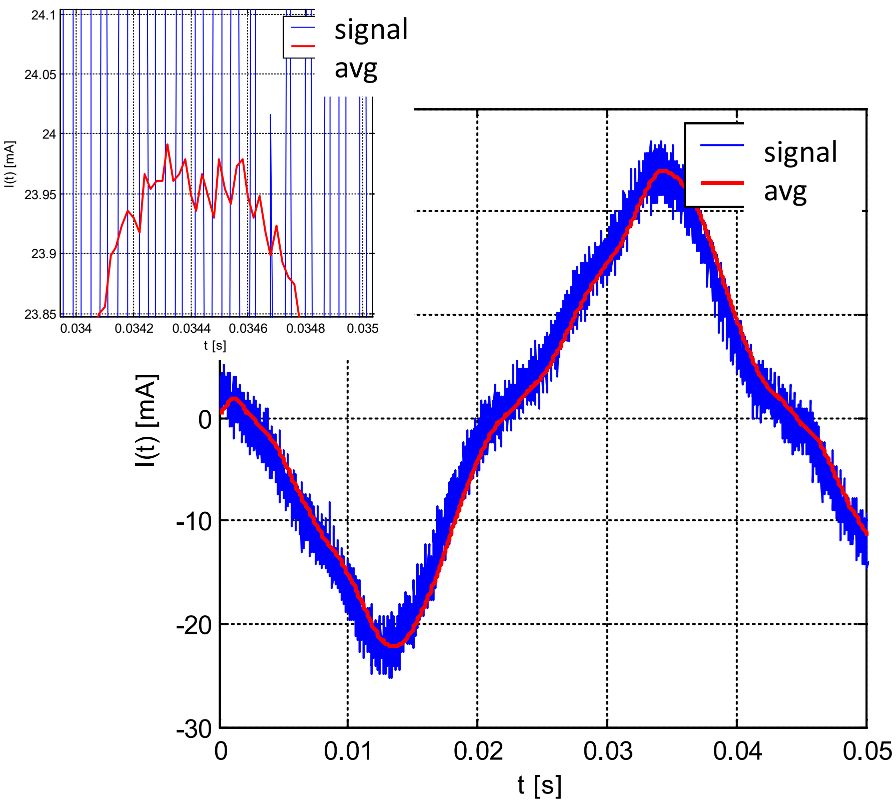
- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- N=32: obvious improvement, level fluctuation approx. 0.1 mA_{pp}

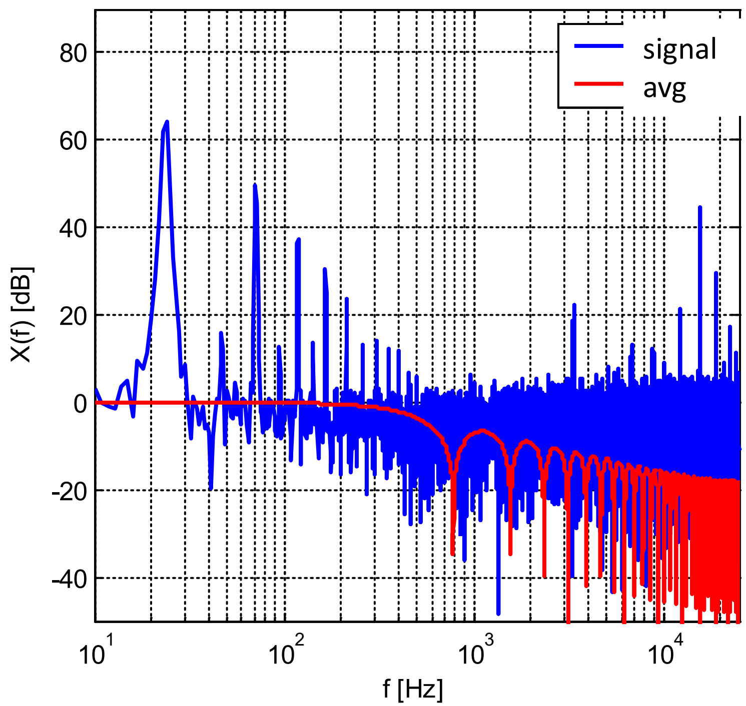




Design: simulation in the time domain

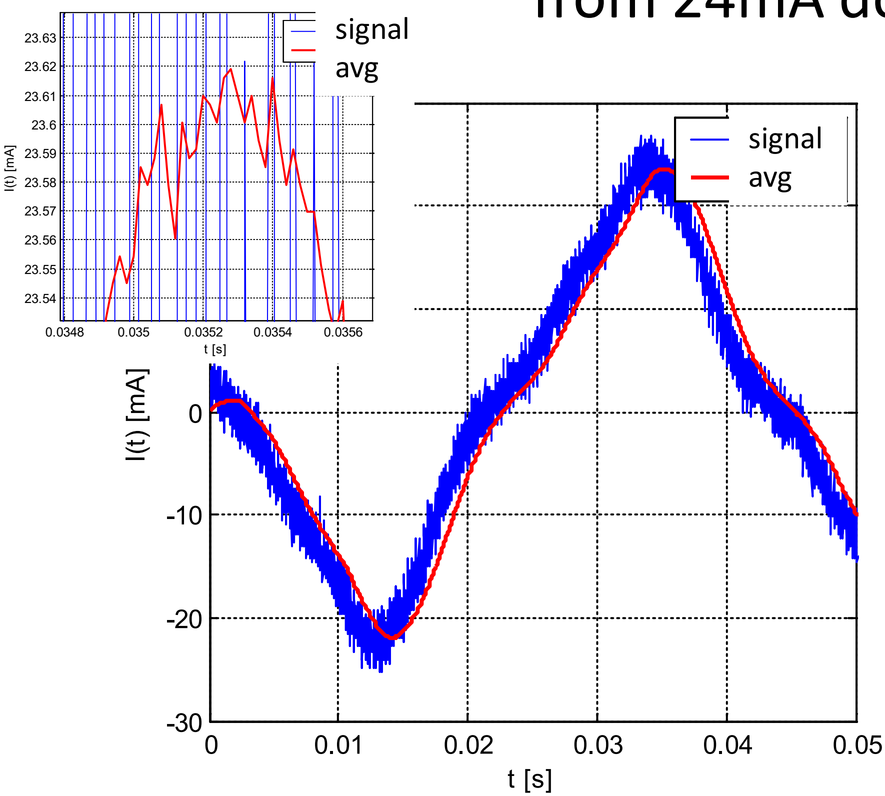
- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- N=64: obvious improvement, level fluctuation <0.1 mA_{pp}

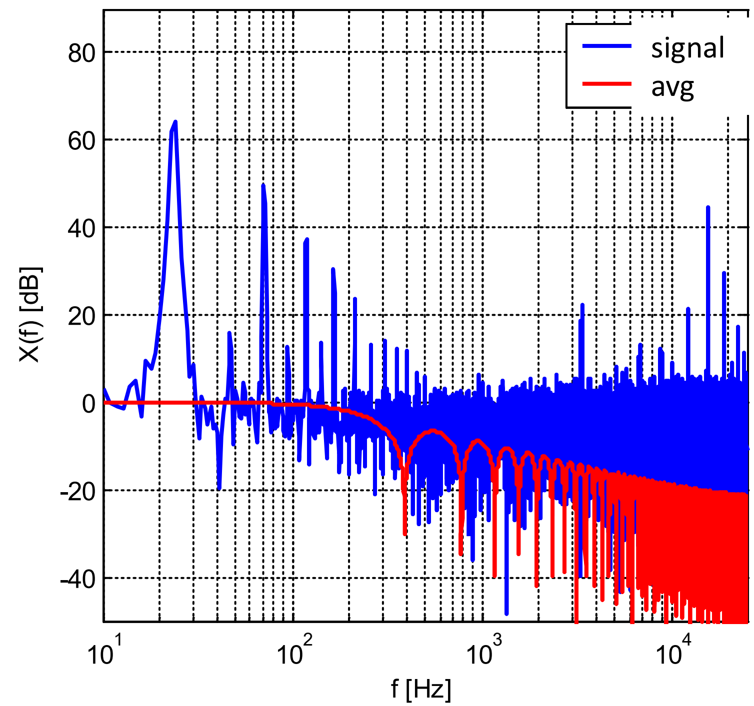




Design: simulation in the time domain

- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- N=128: noise level further reduced as expected, but an increased phase-shift (delay) is observed, the signal amplitude is reduced from 24mA down to 23.6mA: distortion occurs





Implementation

- Sample-wise calling: DAC_data_out = process_movingAv(ADC_data_in);
- Chosen order: N_MOV_AVG=32=0x020 → N_MOV_AVG_MASK = 31 = 0x01F
  - movAv_buff: data-storing buffer
  - movAv_buff_cntr: buffer pointer (actual position)

```
uint32_t process_movingAv(uint32_t data_in){
     int ii;

     // stepping buffer pointer
     movAv_buff_cntr = (movAv_buff_cntr+1)&N_MOV_AVG_MASK ;
     movAv_buff[movAv_buff_cntr] = data_in; // storing data

     // summing
     movAv_sum=0;
     for (ii=0; ii<N_MOV_AVG; ii++){
          movAv_sum += movAv_buff[ii];
     }
     // *1/N ->N_MOV_AVG_BIT;
     data_out_avg = movAv_sum>>N_MOV_AVG_BIT;

     return data_out_avg;
}
```

Implementation: increase speed (recursive)

- Idea: the whole sum is not necessary to be calculated for every sample since it is different by only two terms from the previous sum, so it can be calculated in a recursive manner:
  - $sum_{n+1} = sum_n - x_{n-N} + x_n$
- Example:
  - Data sequence: x1, x2, x3, x4, x5 …
  - y4_sum = x1+x2+x3+x4
  - y5_sum = x2+x3+x4+x5 = y4_sum – x1 + x5
  - Division by 1/N is performed for the whole sum

Implementation: increase speed (recursive)

- What is the transfer function of this version of implementation?
- Time domain:

$$y_n = y_{n-1} + x_n - x_{n-N}$$

- Frequency domain:

$$Y = z^{-1}Y + X - z^{-N}X$$

$$\frac{Y}{X} = \frac{1 - z^{-N}}{1 - z^{-1}}$$

- The transfer function is the same, it is expected!

Implementation: increase speed (recursive)

- Sample-wise calling:

DAC_data_out = process_movingAvFast(ADC_data_in);

```
uint32_t process_movingAvFast(uint32_t data_in){
        // stepping buffer pointer: pointing at the oldest sample: x[n-N]
        // First this data is used to be subtracted from y[n] and store here
        // the newest data
        movAv_buff_cntr = (movAv_buff_cntr+1)&N_MOV_AVG_MASK ;
        // y[n+1] = y[n] + x[n] – x[n-N]
        movAv_sum = movAv_sum + data_in - movAv_buff[movAv_buff_cntr ];
        movAv_buff[movAv_buff_cntr] = data_in; // storing data
        data_out_avg = movAv_sum>>N_MOV_AVG_BIT; // * 1/N
        return data_out_avg;
}
```

Implementation: increase speed (recursive)

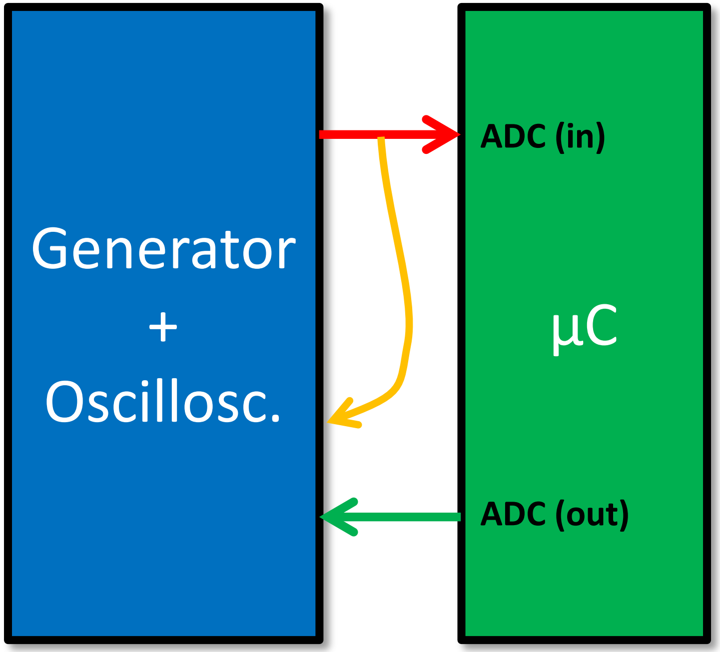
- Warning! The implementation of **speed increment used up to now must not be applied with floating-point numbers** due to number representation limitation. It may happen that the value of the actual sample is out of the representation range of the partial sum. In this case when subtracting the value having been added N steps before the subtraction will not make that value completely null out.
- Example:

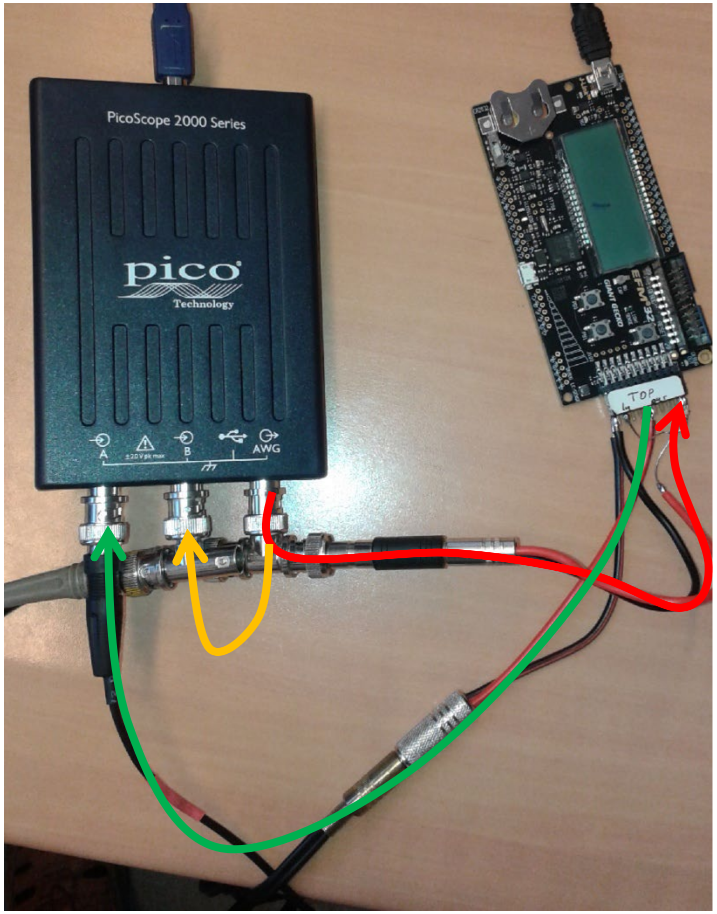
```
float x_array_1[4] = {9, 4.096e4, 9.765625e-2, 6.4e2};
float x_array_2[4] = {9, 4.11,      9.8354,         6.27}; // wrong result for this
float y_1 = x_array_1[0]+x_array_1[1]+x_array_1[2]+x_array_1[3];
y_1 = y_1 - x_array_1[0]-x_array_1[1]-x_array_1[2]-x_array_1[3];

 // wrong result for this despite the fact that the numbers are much closer to each other
float y_2 = x_array_2[0]+x_array_2[1]+x_array_2[2]+x_array_2[3];
y_2 = y_2 - x_array_2[0]-x_array_2[1]-x_array_2[2]-x_array_2[3];
printf("y_1 = %f;    y_2 = %f;\r\n", y_1, y_2);
```

```
y_1 = 0.000000;     y_2 = 0.000001;
```

Measurement setup

- Measurement: USB oszcilloscope and signal generator (PicoScope)
- Signal generator: board connected to ADC and trigger input

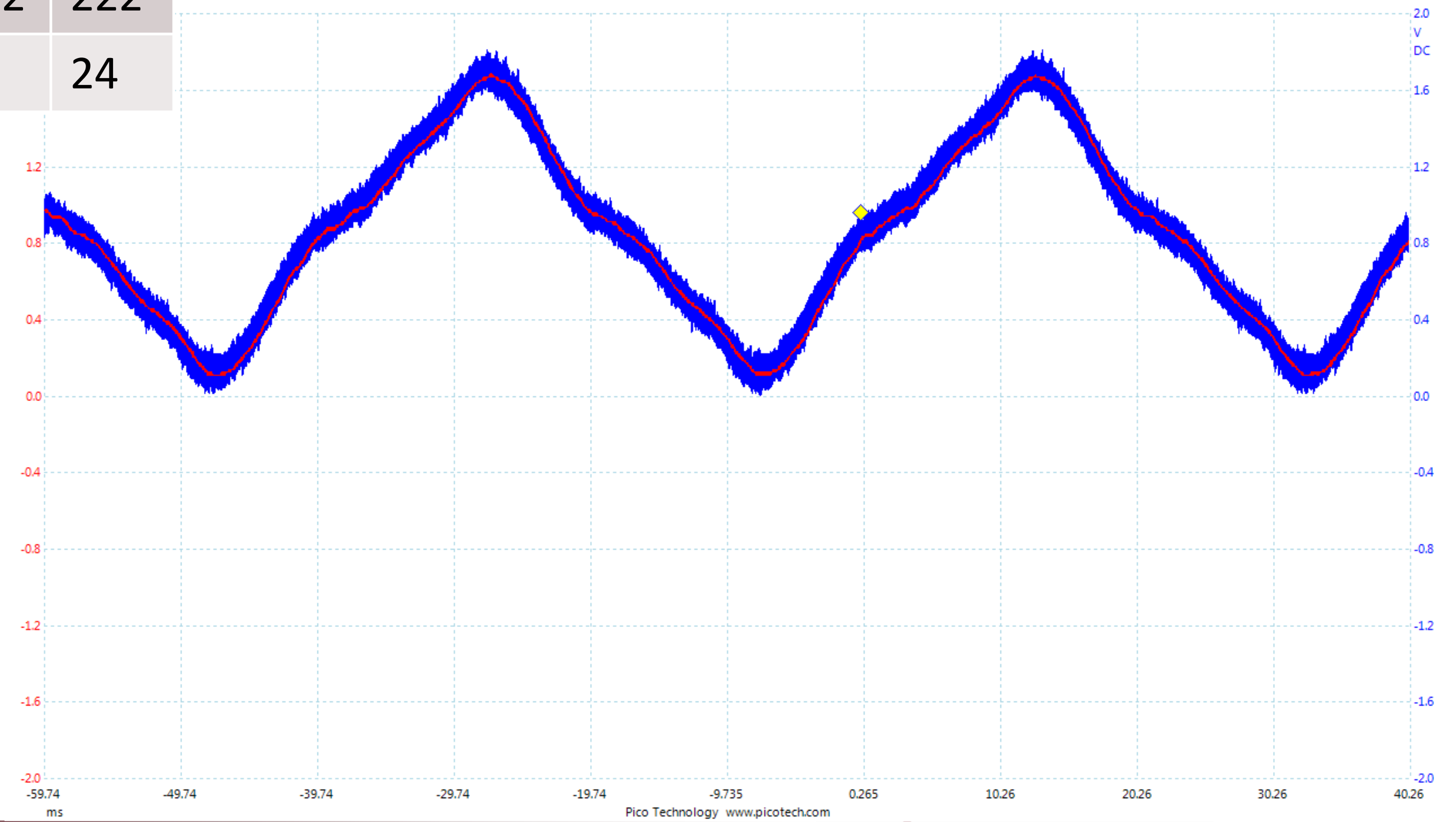




Implementation: measurement result

- In case of both implementations the measurement and simulation results are the same
- Computation need (number of CLK cycles for different optimization levels):

	-O0	-O3
Whole sum	1082	222
Recursive sum	96	24



Moving-average: summary

- Advantages:
  - Small computation need compared to a FIR filter of same order
- Disadvantages
  - The passband is not flat, signal distortion occurs
  - In the stopband the decay is not steep
  - May require large memory
- Design steps:
  - Signal analysis: distinction between useful signal and noise (unwanted signals)
  - Determination of corner frequency (f0): the useful signal components should be under the corner frequency
    - N = fs/f0
  - Performing offline simulation to tune N
    - Trade-off is needed between (i) noise suppression and (ii) signal distortion and delay
- Implementation:
  - In case of fixed-point representation it can be done by one summation, one subtraction and one scaling by N per every sample
  - In case of floating-point representation this implementation method cannot be used

Exponential averaging

Méréstechnika és
Információs Rendszerek
Tanszék

© BME-MIT

Exponential averaging

- A drawback of moving-average the potentially large memory need: N samples to be stored. N as order can be especially high when sampling frequency is high and bandwidth is narrow.

- Exponential averaging:

$$y_n = y_{n-1} + (1-\alpha)(x_n - y_{n-1})$$

- Other form (same meaning):

$$y_n = \alpha\, y_{n-1} + (1-\alpha)\, x_n$$

- Exponential averaging keeps only the low-frequency components of the signal
- Parameter α is a constant value close to 1

Exponential averaging

- Intuitive explanation:

$$y_n = y_{n-1} + (1-\alpha)(x_n - y_{n-1})$$

  - Variable y contains the low-frequency components. Considering the input as a noisy DC voltage, then **y is the estimator of the DC (average) voltage**.

    new estimated value = old estimated value + weighting * (actual sample –old estimated value)

    - When estimated value is smaller than the real one then the estimated value is increasing toward the real value until reaching the real value in average. At this time instant $(x_n - y_n)$ expression becomes zero
    - Expression $(x_n - y_n)$ can be considered as an error of the estimation
    - The smaller α the larger weight is applied on the estimation error: faster settling but more sensitive to disturbances

$$y_n = \alpha\, y_{n-1} + (1-\alpha)\, x_n$$

  - In this form the new estimated value is given as the weighted sum of the old estimated value and actual sample. The sum of the weights is 1.
    - The smaller α the larger weight is applied on the new input sample. Settling is faster but more sensitive to noise.

Transfer function

- System equation:

$$y_n = \alpha\, y_{n-1} + (1-\alpha)\, x_n$$

- Calculation of transfer function (f: frequency, fs: sampling frequency)

$$Y = \alpha\, z^{-1} Y + (1-\alpha)\, X$$

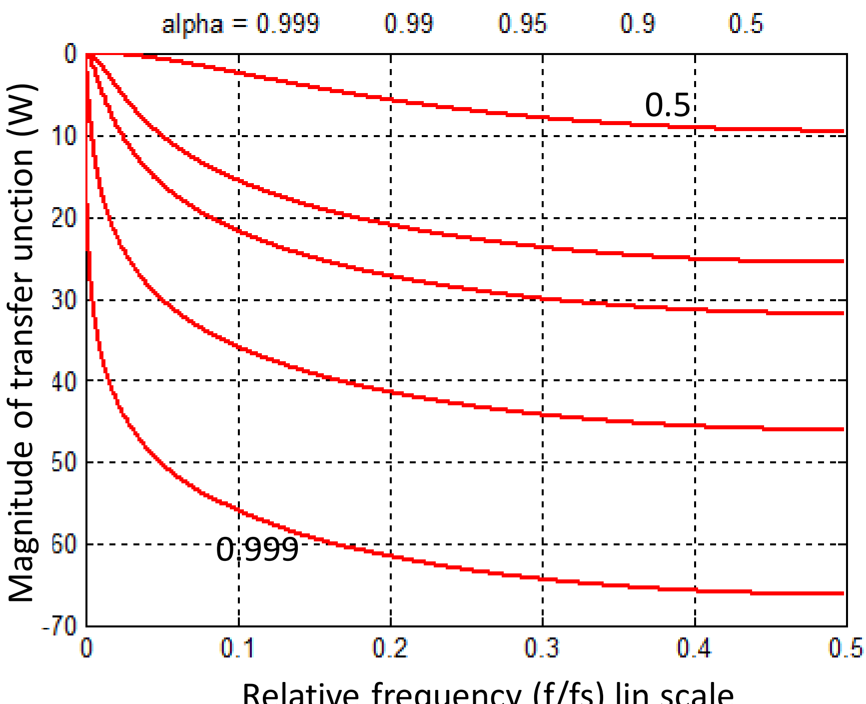
$$Y - \alpha\, z^{-1} Y = (1-\alpha)\, X$$

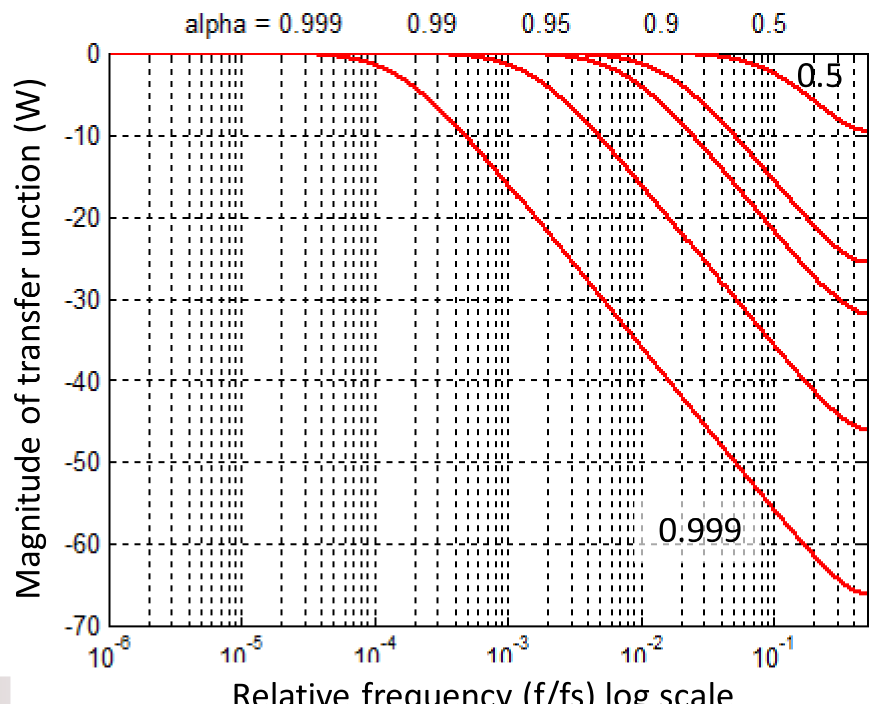
$$W(z) = \frac{Y}{X} = \frac{1-\alpha}{1-\alpha\, z^{-1}}$$

$$W(z) = \frac{1-\alpha}{1-\alpha\, e^{-j2\pi\frac{f}{fs}}}$$

Transfer function

- Transfer function: $W(z) = \dfrac{1-\alpha}{1-\alpha z^{-1}} = \dfrac{1-\alpha}{1-\alpha e^{-j2\pi\frac{f}{fs}}}$
- The closer alpha to 1:
  - the smaller the bandwidth
  - the larger the suppression at the certain frequency
  - the larger the settling time (more time is needed to settle)





Time-domain characteristic

- Calculation of impulse response. System eq.: $y_n = \alpha\, y_{n-1} + (1-\alpha)\, x_n$
- Let it be x_0=1 otherwise x_n=0 constant (this is the excitation impuse), then:

$$y_0 = (1-\alpha)$$

$$y_n = \alpha\, y_{n-1} + (1-\alpha)\, 0$$

- Therefore:

$$y_0 = (1-\alpha)$$

$$y_1 = \alpha\, y_0 = \alpha\,(1-\alpha)$$

$$y_2 = \alpha\, y_1 = \alpha^2\,(1-\alpha)$$

…

$$y_n = \alpha^n\,(1-\alpha) = \alpha^n\, y_0$$

Time-domain characteristic

- What is time constant (impulse response reaches its initial value*1/e)?

$$y_n = \alpha^n (1-\alpha) = \alpha^n y_0$$

- So what n=N makes true that:

$$\alpha^N = 1/e = e^{-1}$$

- Take the e-base logarithm of both sides:

$$N \ln(\alpha) = \ln(e^{-1}) = -1 \longrightarrow \ln(\alpha) = \frac{-1}{N}$$

- Raise e (back) to the power of the two sides:

$$e^{\ln(\alpha)} = e^{\frac{-1}{N}}$$

$$\alpha = e^{\frac{-1}{N}}$$

- So if the sampling frequency is fs and time constant is T, then time constant written as the number of samples results in $N = f_s \cdot T$
- So parameter alpha:

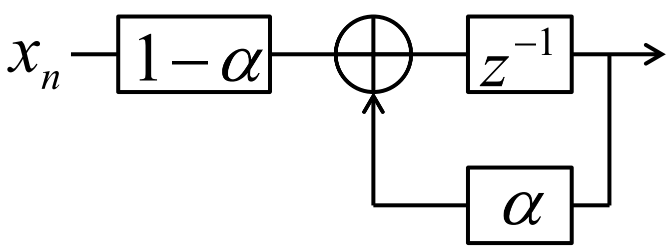
$$\alpha = e^{\frac{-1}{N}} = e^{\frac{-1}{f_s \cdot T}}$$

Time-domain characteristic

- How the impulse response look like? (after the substitution of alpha)

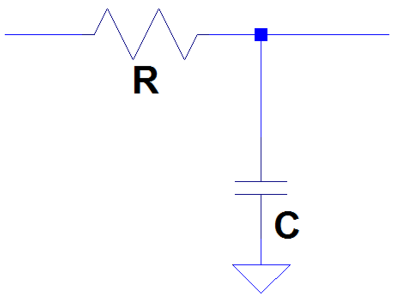
$$y_n = \alpha^n (1-\alpha) = (1-\alpha)\cdot e^{-\frac{n}{f_s \cdot T}} = (1-\alpha)\cdot e^{-\frac{n/f_s}{T}} = (1-\alpha)\cdot e^{-\frac{n\cdot T_s}{T}}$$

- Where Ts=1/fs the sampling time period
- Note, t=n*Ts so in the nominator of the power the real time of n-th step can be found
- It can be clearly seen that the impulse response is decreasing exponentially by T time constant
- Analogy: exponential averaging is the discrete-time simulation of a low-pass R-C filter



$$y_n = \alpha\, y_{n-1} + (1-\alpha)\, x_n$$

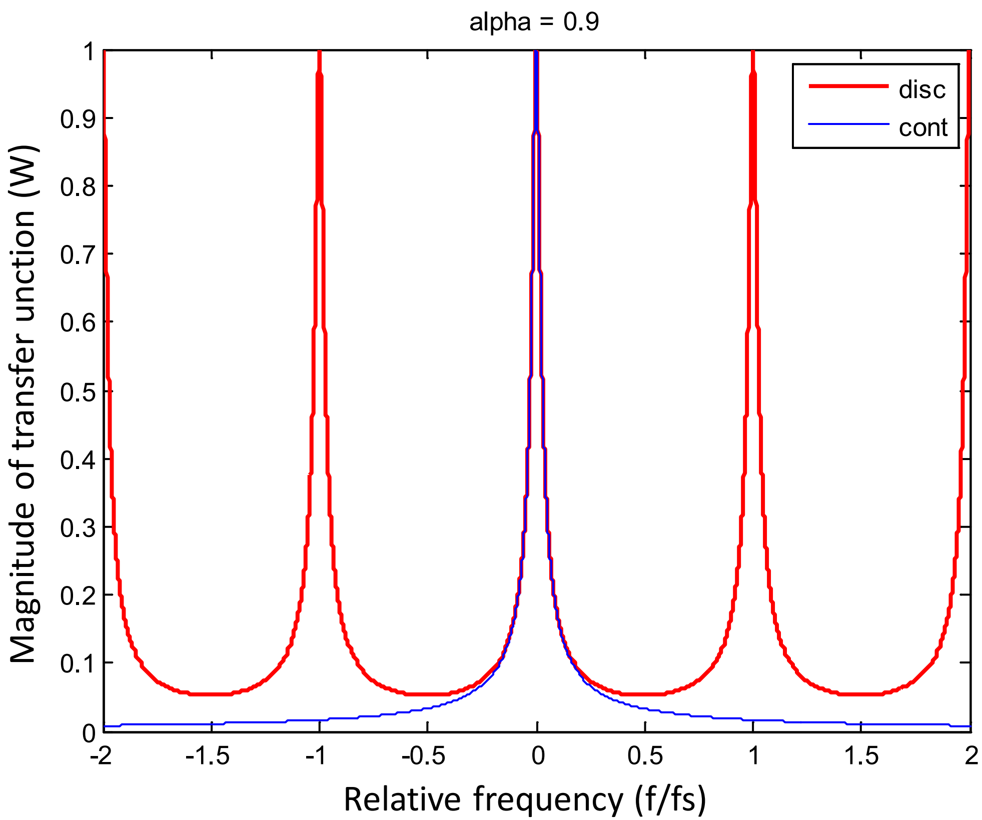
$$\alpha = e^{\frac{-1}{f_s \cdot RC}}$$



$$T = RC$$

Analogy

- The transfer function of the discrete-time exponential averaging can be well estimated by a the transfer function of an R-C-in-series system. The estimation error is very small when cutoff frequency<<sampling
- Can be considered as if the continuous-time transfer function had been repeated at every fs and summed up.



$$W(z) = \frac{1-\alpha}{1-\alpha e^{-j2\pi\frac{f}{fs}}}$$

$$W(s) = \frac{1}{1+jfT}$$

$$\alpha = e^{\frac{-1}{f_s \cdot T}}$$

Design procedure

- Design procedure is the same as shown for moving average design example
- Recall:
  - Data acquisition
  - Spectrum evaluation
  - Distinction signal from noise

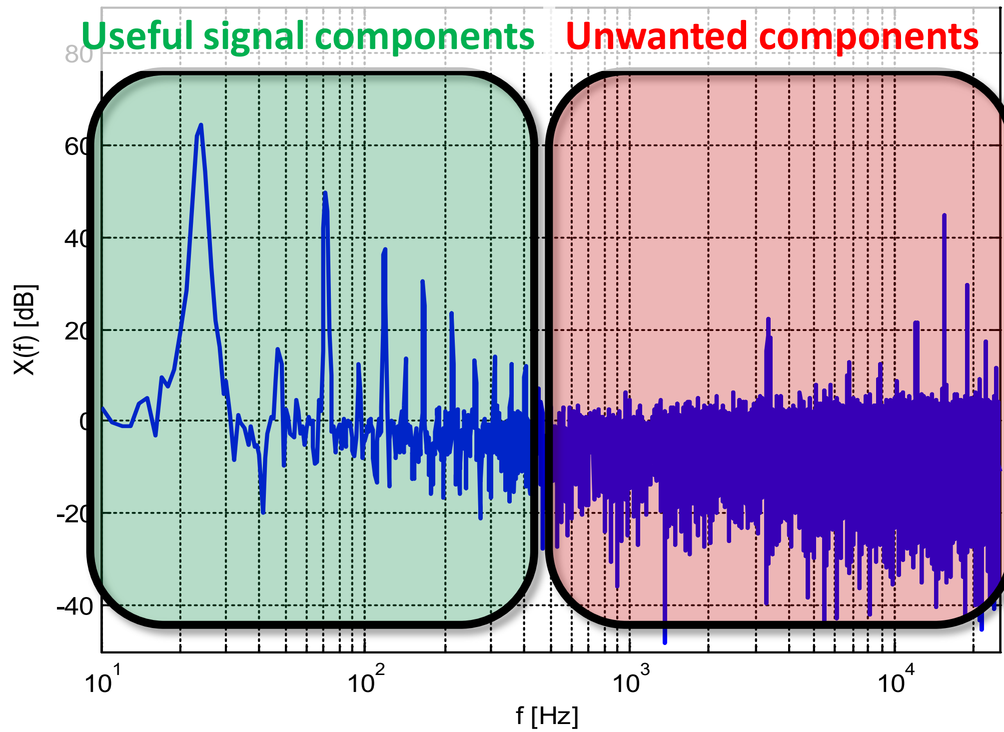
- **Sampling freq.: fs=50kHz**
- **Cutoff freq.: 500Hz**
- **Time const.: T=1/(2*π*500Hz)**

  T=0.32ms
- **Calculation of alpha param.:**

$$\alpha = e^{\frac{-1}{N}} = e^{\frac{-1}{f_s \cdot T}}$$

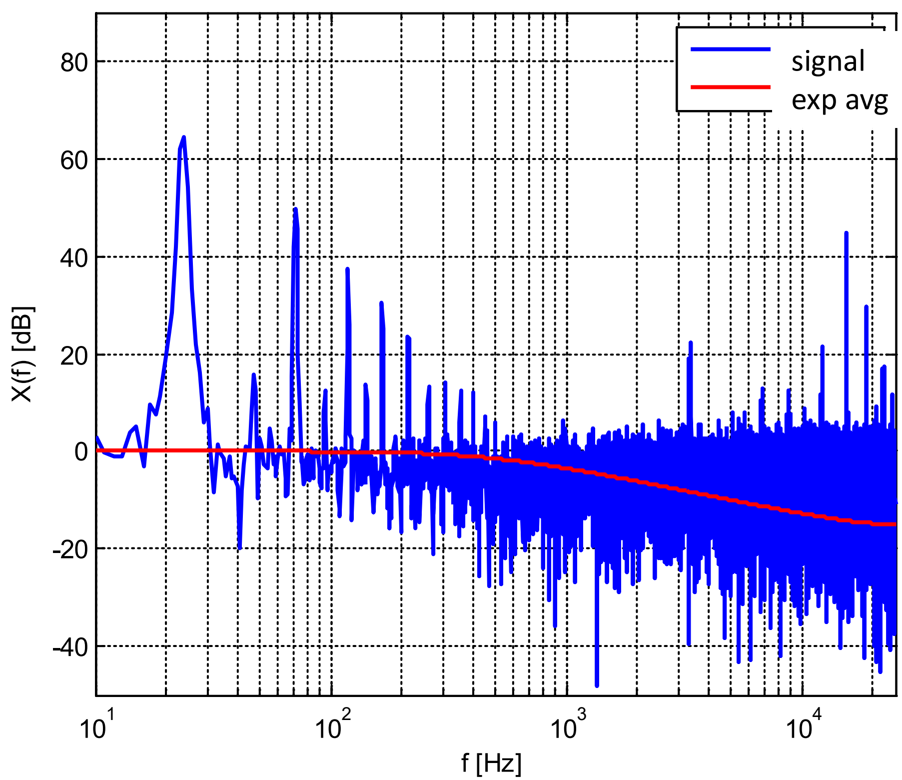
$$N = T \cdot f_s = 032ms \cdot 50kHz = 15.9$$

$$\alpha = e^{\frac{-1}{50kHz \cdot 0.32ms}} = 0.9391$$



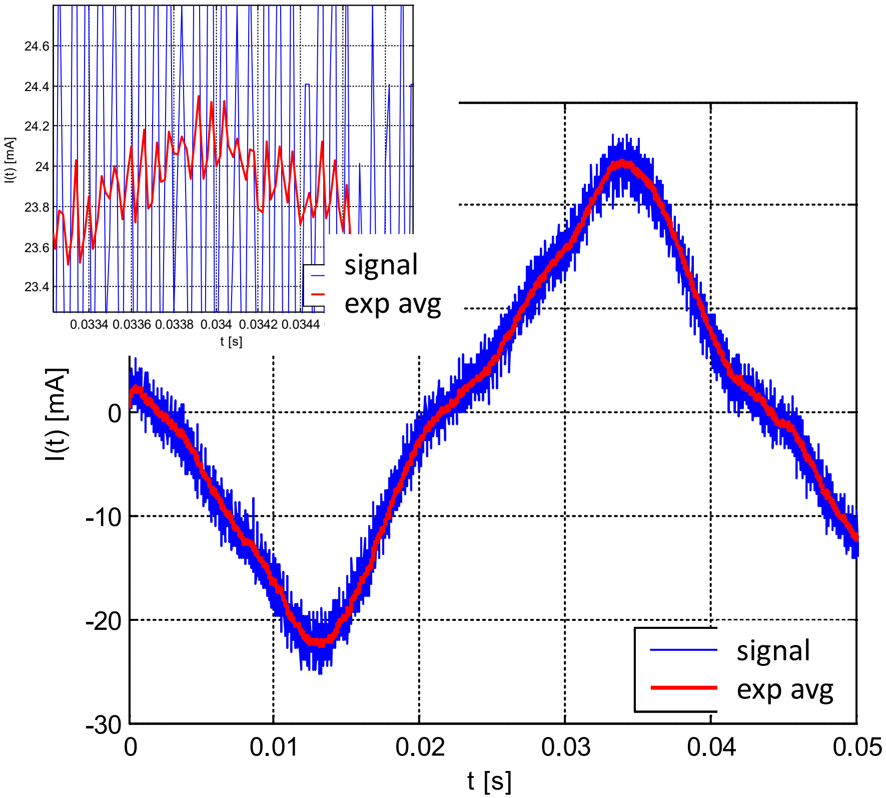
Design: checking in the frequency domain

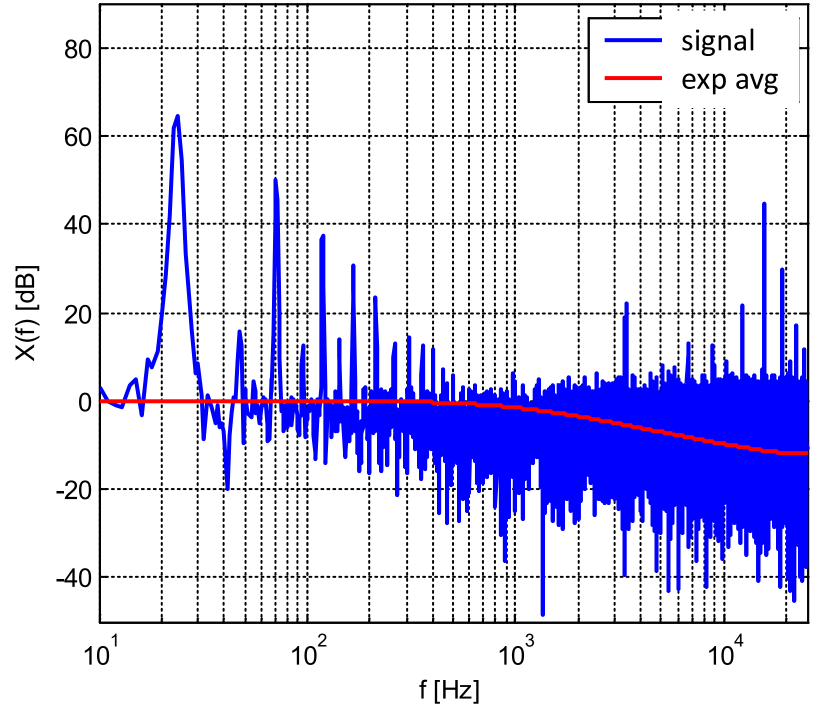
- Let the cut-off frequency be approx. 500Hz.
- Checking the transfer function of the designed averaging procedure
- Figure shows signal spectrum (blue) and the transfer function of averaging (red)
- Averaging slightly distorts the useful signal: check the signal in the time domain, to see the distortion whether it is acceptable or not



Design: simulation in the time domain

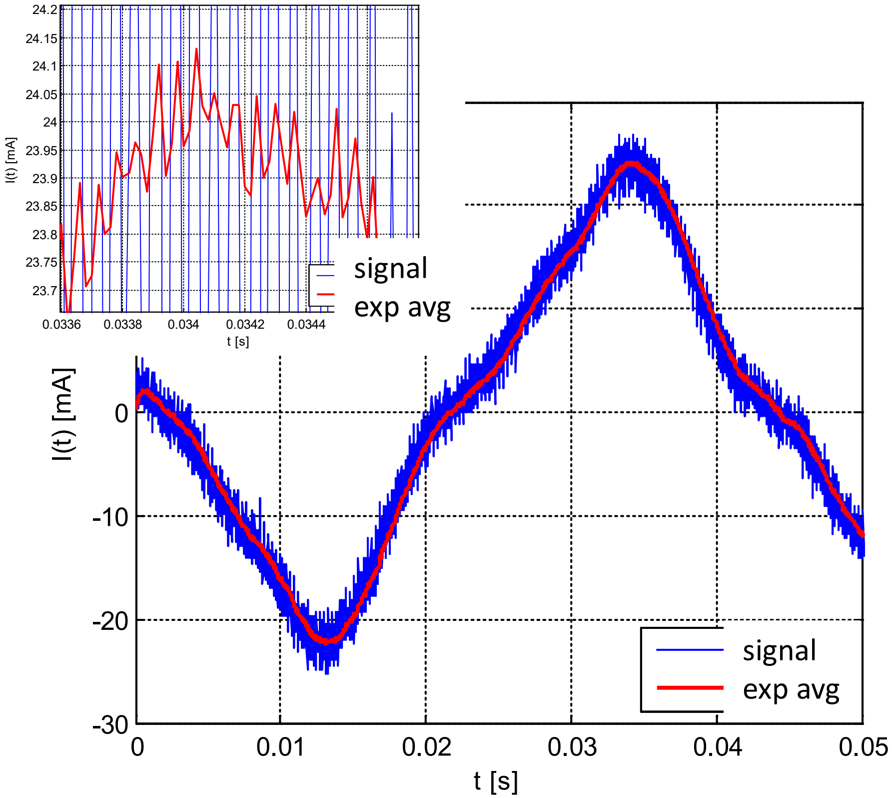
- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- fc=1000Hz: slight improvement, but the signal is still noisy (approx. 0.4 mA_{pp})

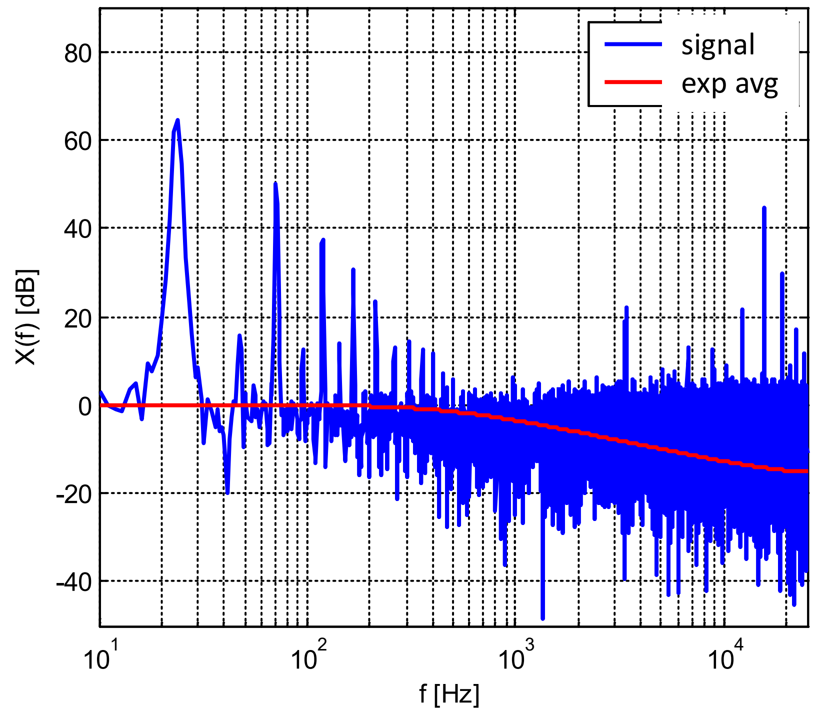




Design: simulation in the time domain

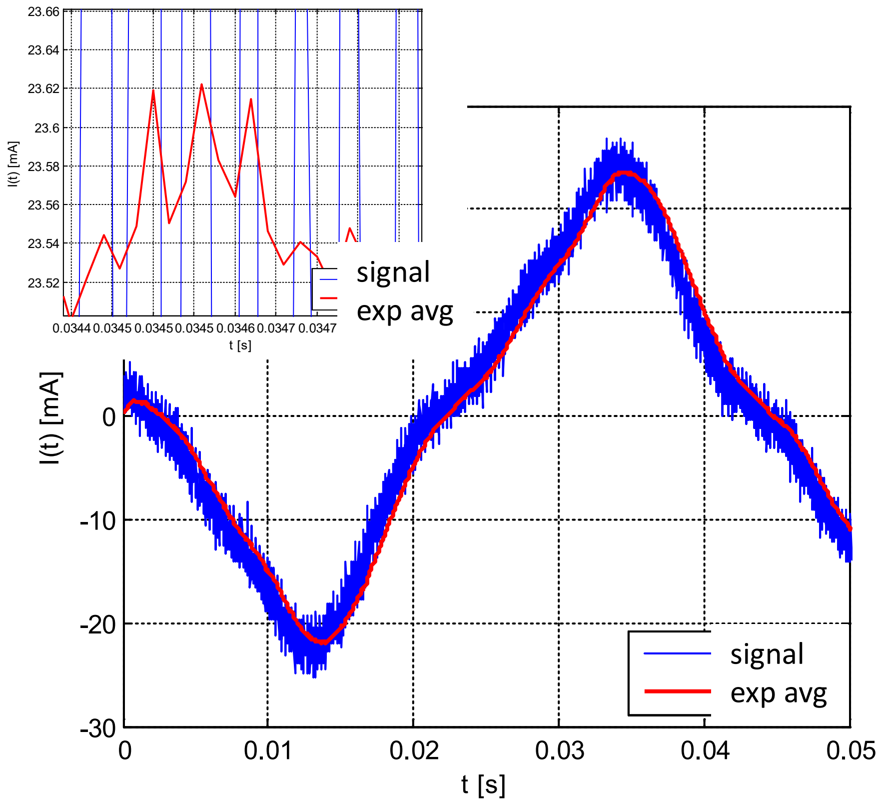
- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- fc=500Hz: considerable improvement, noise further reduced (approx. 0.15 mA_{pp})

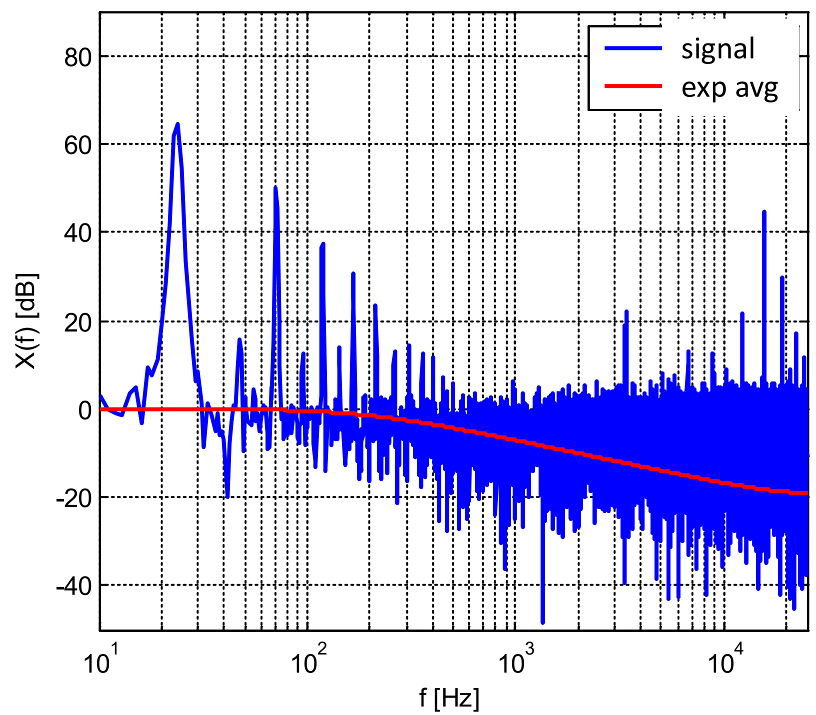




Design: simulation in the time domain

- Checking by time-domain simulation how signal-to-noise ratio and signal distortion is affected by the order number
- fc=200Hz: noise level further reduced as expected (<0.1mA), but an increased phase-shift (delay) is observed, the signal amplitude is reduced from 24mA down to 23.6mA: distortion occurs





Implementation

- Sample-wise calling:

DAC_data_out = process_Filter(ADC_data_in);

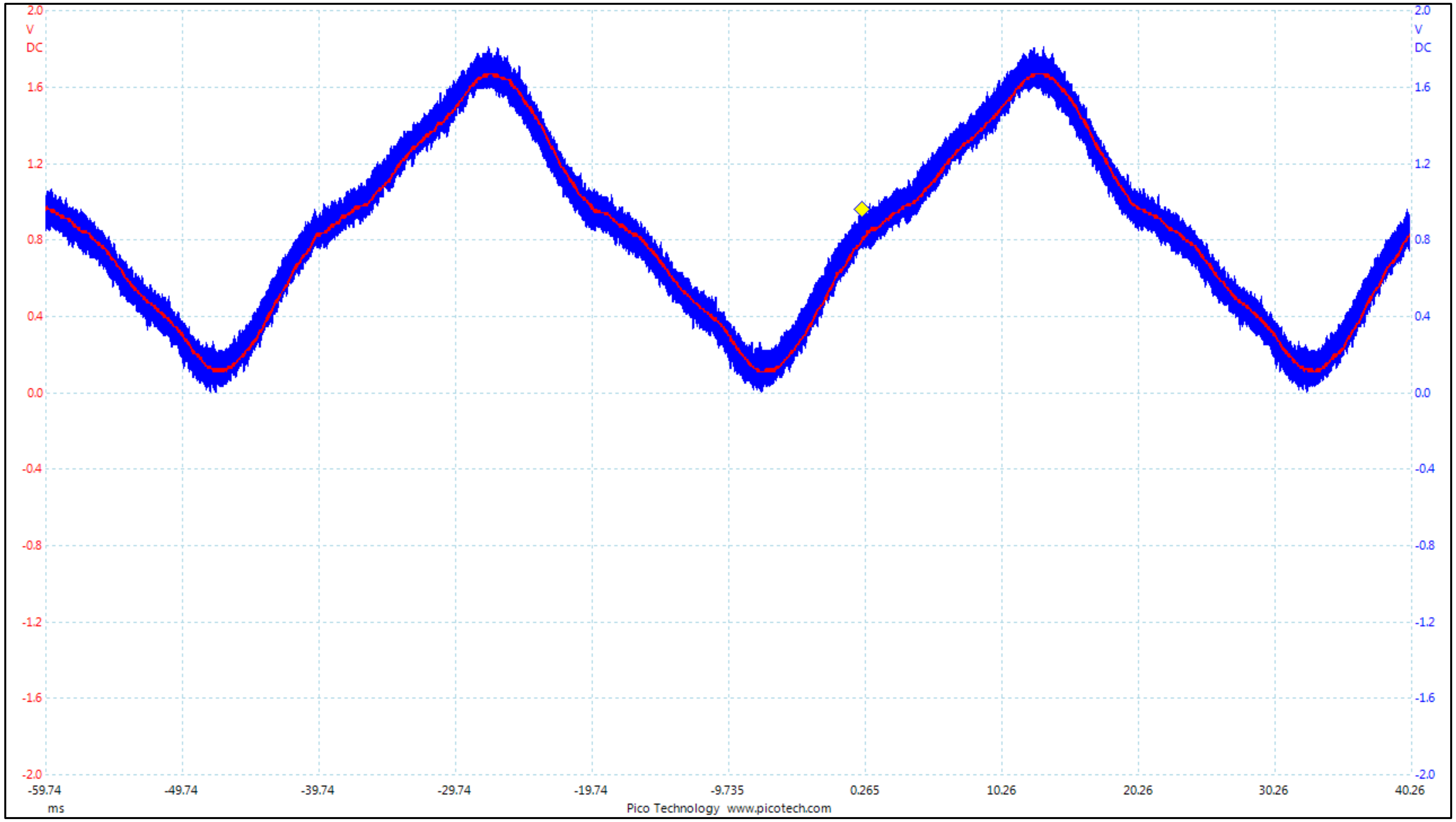
```
uint32_t process_Filter(uint32_t data_in)
{
        uint32_t data_out;
        float data_in_f;
        float alpha = (1- 0.9391); // time constant
        static float y;

        data_in_f = (float)data_in; // conversion into floating point

        y = y + alpha*(data_in_f - y); // algorithm of exponential averaging

        data_out = (uint32_t) y; // conversion into fixed-point

        return data_out;
}
```

State variable *y* must preserve its value even among function calls therefore *static* is needed

© BME-MIT

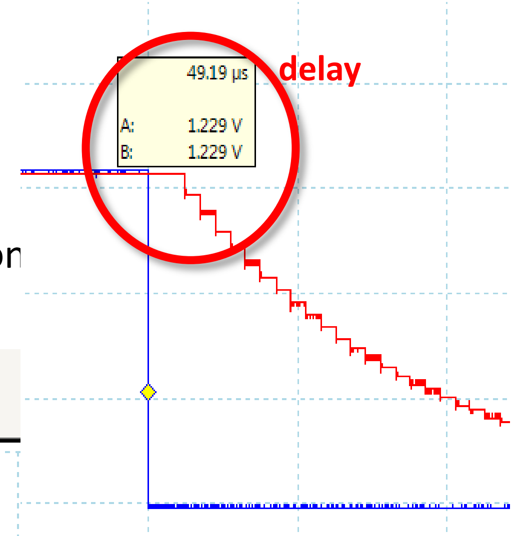
Measurement results

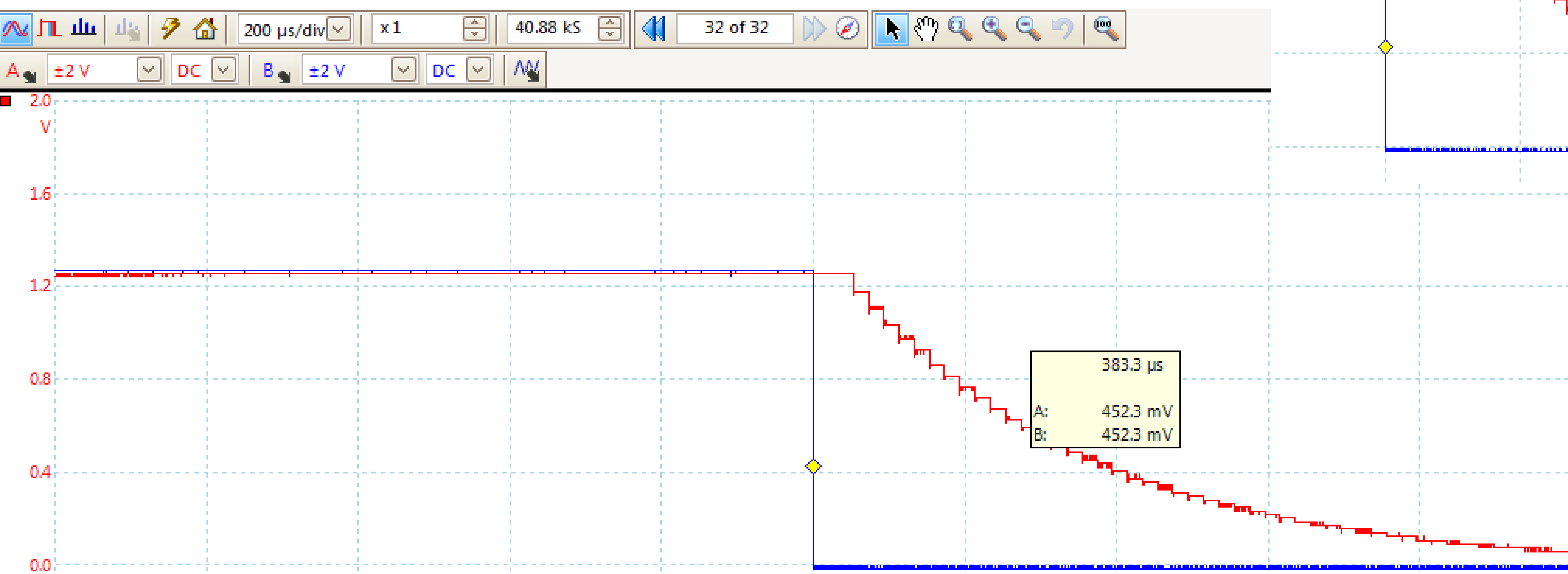
- In case of both implementations the measurement and simulation results are the same
- Implementation: floating-point
  - Efficiency is not that good (better methods exists)



Measurement results

- Step-response (square-wave excitation, fs=50kHz, 1.23V peak)
- Time constant: 1.23V/e=0.4525V reached after settling time
- Time constants:
  - In theory: T=1/(2*pi*500 Hz)=318 µs (318 µs*50kHz≈16 minta)
  - Measured: 383 µs – 40 µs=343 µs (measured value-delay)
  - Measured and theoretical time constants are very close
    - Sampling time 20µs (50 kHz), so we are in the order of quantization error





Exponential averaging: summary

- Advantages:
  - Low computation need compared to a high order FIR filter
  - Low memory need
- Disadvantages
  - Fractional numbers are needed to be used
  - Transient region of transfer function is not steep
- Design steps:
  - Signal analysis: distinction of signal from noise and disturbances
  - Determination of corner frequency (fc) and from fc a time constant (T=1/fc/2/pi): useful signal components should be below fc
  - Determination of parameter alpha from time constant: $\alpha = e^{\frac{-1}{N}} = e^{\frac{-1}{f_s \cdot T}}$
  - Offline simulation, tuning of alpha: trade-off between (i) noise reduction and (ii) signal distortion and delay
- Implementation:
  - Fractional numbers are needed to be used
    - Floating point is not that important to be used, discussed maybe later…

© BME-MIT

Moving-average and exp. averaging

- Plotting time function and spectrum
- Transfer functions of the filters fit well on each other due to similar design principles
- Time functions fit well on each other, not really possible to make a distinction
- Relation: applying the following parameters the two methods results in nearly the same noise reduction: $N = \frac{2}{1-\alpha} - 1$
  - example :

Recall:
in the case of moving-average

$N \approx \frac{2}{1-\alpha} - 1 = \frac{2}{1-0.9391} - 1 = 31.84$ ---> N=32 was applied