Concepts and Software Design for CPS

Lab 3: Digital Filtering

Raphael Trumpp   Binqi Sun

Chair of Cyber-Physical Systems in Production Engineering
Technical University of Munich
Munich, Germany

Lab 3: Overview

MATLAB

Signal Processing

Digital Filters

Assignment 3 (due: 25.11.2021)

MATLAB

MATLAB

MATLAB (matrix laboratory) is a numerical computing environment.

It uses its own programming language for numerical calculations in scripts. In MATLAB all standard operations (like multiplication) and most functions (like `sine(...)`) are also available on vectors and/or matrices.

We will use MATLAB in this Lab to investigate digital filters.

MATLAB
Installation

As a TUM student, you are allowed to download and use Matlab:

https://wiki.rbg.tum.de/Informatik/Helpdesk/MatlabInstallieren

In the case of low-level graphics issues, start *Matlab* with *OpenGL*:

```
$ matlab -softwareopengl
```

Signal Processing

Signals in Time Domain

Creating a Signal

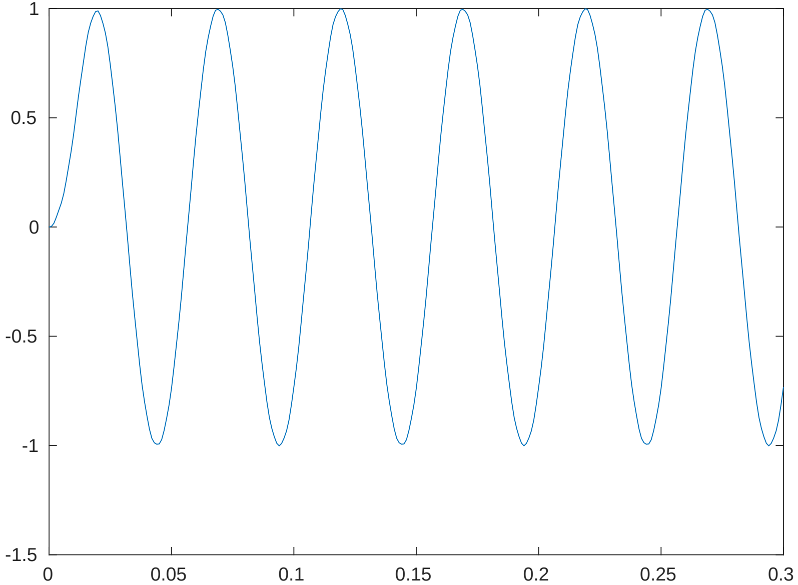
Signals can be created by applying functions to an array of input values:

```
% Sample t at f_s=1000Hz for 0.3sec
f_s = 1000;
t = 0:1/f_s:0.3; % values from 0 to 10 in steps of 0.001
% Generation of sample sin wave
phase_sin = 0;
amplitude_sin = 1;
f_sin = 20; % 20Hz base signal
omega_sin = 2*pi*f_sin;
signal_sin = amplitude_sin * sin(omega_sin*t + phase_sin);
```

This example creates a sine wave signal. We can plot this signal using `clf; plot(t,signal_sin)` as can be seen on the next slide.

Signals in Time Domain

Creating a Signal (Plot)



Signals in Time Domain

Overlaying Signals

We can also add signals:

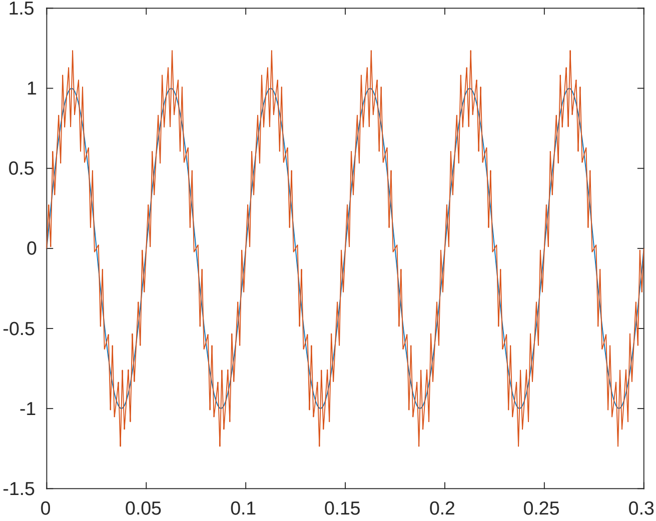
```
% Generation of noise signal
phase_noise = 0;
amplitude_noise = 0.25;
f_noise = 400; % 400Hz noise signal
omega_noise = 2*pi*f_noise;
signal_noise = amplitude_noise*sin(omega_noise*t + phase_noise);

% Combine sin signal with noise
signal_combined = signal_sin + signal_noise;
```

This example creates a signal composed of two sine waves (carrier and noise). The result can be seen on the next slide.

```
figure(2)
plot(t, signal_sin);
hold on
plot(t, signal_combined);.
```

Signals in Time Domain
Overlaying Signals (Plot)



Signals in Frequency Domain
Fourier Transform

We use the Fourier Transform to transform the signal to the frequency domain.

```
% FFT analysis
figure(3)
signalFT = fft(signal_combined(1:end-1)); % odd sized signal: (1:
    end-1)
signalFT_amplitude = abs(signalFT/length(signalFT));
signalFT_window = signalFT_amplitude(1:length(signalFT)/2+1);
signalFT_window(2:end-1) = 2*signalFT_window(2:end-1);
frequencies = f_s*(0:(length(signalFT)/2))/length(signalFT);
plot(frequencies, signalFT_window);
```

The result can be seen on the next slide.

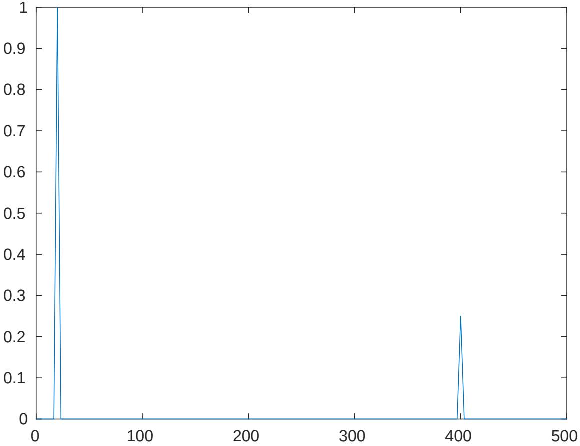
For more information please refer to:

https://de.mathworks.com/help/matlab/ref/fft.html.

Signals in Frequency Domain

Fourier Transform (Plot)

We can see the signal as two sine waves of 20 Hz and 400 Hz.



To export generated signals to a file, or import signals from a file, you can use these functions:

```
csvwrite("filename.csv", signal.');

signal = csvread("filename.csv").';

dlmwrite("filename.csv",signal.','precision','%.6f');
```

The operator `.'` transposes a matrix/vector to make a row vector of values to a file with one value per line and vice versa.

We will use these functions to write generated signals to a file, that can then be read from our C program. The other way around, we can validate our C programs output values by writing them to a file and reading in the file to MATLAB.

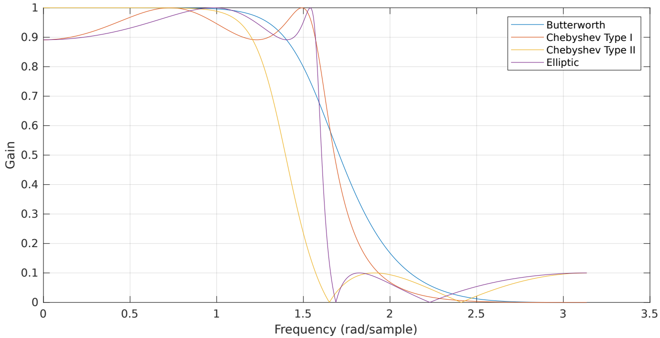
Digital Filters

Digital Filters

Filter Type

Low-pass filters are often used in cyber-physical systems to reject high frequency noise *e.g.*, from power conversion modules.

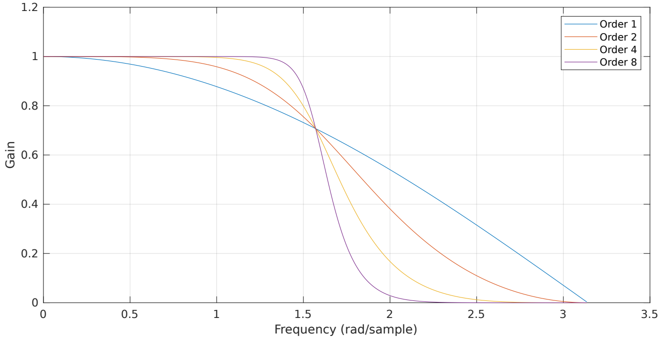
There are different types of filters that can be applied. They vary in their characteristics gain (see below) and phase shift per frequency.



Digital Filters

Filter Order

Every filter has parameter called *Order*, which describes how strong the gain changes between the pass (ideally gain $== 1$) and the stop frequencies (ideally gain $== 0$).
One of the basic filters is the Butterworth filter, which we will be using for our Lab. Below you can see the gain over the frequency of a Butterworth filter from Order 1 to 4.



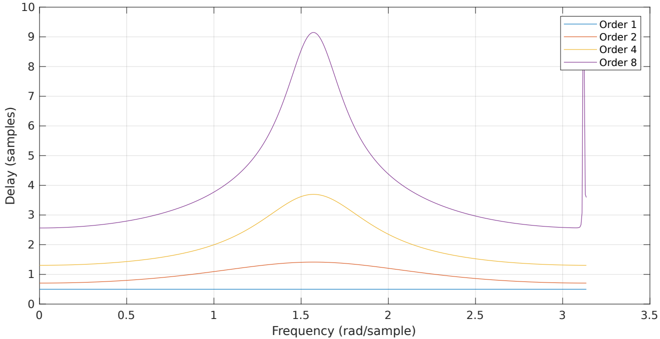
Additional to the order, all filters have another parameter called the *Cutoff Frequency*. In the previous figure you can see a point where all 4 curves cross (here PI/2 because the plot is normalized). The frequency where this happens for Butterworth filters is the cutoff frequency (gain == -3 dB).

Digital Filters

Group Delay

Filters also change the signal in time. This can be summarized in the *Group Delay*: The frequency dependent delay of signal changes.

Below you can see the group delay (in samples over normalized frequency) for a Butterworth filter of order 1 to 8.



[B,A] = **butter**(N,Wn) designs an Nth order lowpass digital Butterworth filter and returns the filter coefficients in length N+1 vectors B (numerator) and A (denominator). The cutoff frequency Wn must be $0.0 < Wn < 1.0$, with 1.0 corresponding to half the sample rate.

[H,W] = **freqz**(B,A,N) returns the N-point complex frequency response vector H and the N-point frequency vector W in radians/sample of the filter with coefficients A and B.

Y = **filter**(B,A,X) filters the data in vector X with the filter described by vectors A and B to create the filtered data Y.

Y = **db2mag**(YDB) computes Y such that YDB = 20*log10(Y).

YDB = **mag2db**(Y) converts magnitude data Y into dB values.

Tip: Use Matlab *help* command. For example: >> help butter

Class exercise

- The frequencies of interest in the signal are from 10 to 60 Hz. There are serious noises with frequencies in the range 500-1000 Hz.
- Suppose that we pick cut-off frequency at 100 Hz, what should be the order (stages) of the filter so that the signal will be reduced no more than 30% while the noise will be reduced at least 96%?

$$|H(\omega)| = \left(\frac{1}{\sqrt{1 + \left(\frac{\omega}{\omega_{cutoff}} \right)^2}} \right)^N$$

Class exercise

- The frequencies of interest in the signal are from 10 to 60 Hz. There are serious noises with frequencies in the range 500-1000 Hz.
- Suppose that we pick cut-off frequency at 100 Hz, what should be the order (stages) of the filter so that the signal will be reduced no more than 30% while the noise will be reduced at least 96%?

$$\left(\frac{1}{\sqrt{1+\left(\frac{60}{100}\right)^2}}\right)^2 = 0.74, \quad \left(\frac{1}{\sqrt{1+\left(\frac{500}{100}\right)^2}}\right)^2 = 0.038$$

Class exercise

- The frequencies of interest in the signal are from 10 to 60 Hz. There are serious noises with frequencies in the range 500-1000 Hz.
- Suppose that we pick cut-off frequency at 100 Hz, what should be the order (stages) of the filter so that the signal will be reduced no more than 30% while the noise will be reduced at least 96%?

In Matlab, you can get the filter coefficients using function **butter**

```
[b,a] = butter(2,100/500)    % wn=1000/2
b =
  0.0675 0.1349 0.0675
a =
  1.0000 -1.1430 0.4128
```

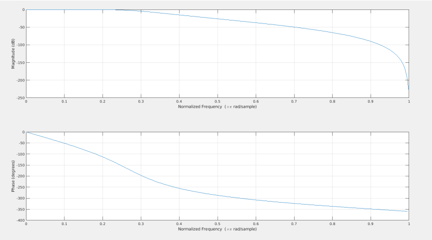
Digital Filters

Filter Design in Matlab

Class exercise

- The frequencies of interest in the signal are from 10 to 60 Hz. There are serious noises with frequencies in the range 500-1000 Hz.
- Suppose that we pick cut-off frequency at 100 Hz, what should be the order (stages) of the filter so that the signal will be reduced no more than 30% while the noise will be reduced at least 96%?

You can plot the frequency response using **freqz**(b,a)



You can try to increase the filter order and observe how varies the phase response.

For data sampled at 1000 Hz, design a low-pass filter with less than 3 dB of ripple in the passband, defined from 0 to 40 Hz, and at least 60 dB of attenuation in the stopband, defined from 150 Hz to the Nyquist frequency (500 Hz).

```
Wp = 40/500; Ws = 150/500;
[n,Wn] = buttord(Wp,Ws,3,60)
n =
     5
Wn =
    0.0810
[b,a] = butter(n,Wn);
freqz(b,a,512,1000); title('n=5 Butterworth Lowpass Filter')
```

Assignment 3 (due: 25.11.2021)

Task 1: Digital Filter in C

Write a generic filter in C (refer to Lecture 3).

Your implementation should:

- use the data type float,
- read the sampled input signal from a file,
- write the output signal data to a file and
- be generic by order N (`#define FILTER_ORDER ...`) and coefficients A, B (e.g., `float A[FILTER_ORDER+1] = ...`),

Your implementation should not:

- compute the filter coefficients (these should be hardcoded)

Note: If necessary you can assume a maximum filter order of 8.

All information needed for implementation is in Lecture 3.

Task 2: Filter Application

For data sampled at 1000 Hz, design a low-pass filter with less than 3 dB of ripple in the passband, defined from 0 to 40 Hz, and at least 40 dB of attenuation in the stopband, defined from 200 Hz to the Nyquist frequency. The filter order should be less than 6.

- You can use the matlab function.
- Find a solution with the lowest filter order.
- Plot the frequency response.
- Generate a sample sine wave and a noise sine signal with the same amplitudes. Add both signals.
- Check if your filter removes the noise form the signal.

Use the C filter from the Task 1 to remove the noise.

- Export the signal with the noise generated in Matlab to a file.
- Run your C filter.
- Import the output of the C-filter to Matlab and plot it.

Next Lab: Lab 4 on 25.11.2021

In 2 weeks: Q&A Meetings!

Next Lab: Lab 4 on 25.11.2021
Review Meetings!

- Topic: PID Control
- Feedback meeting: Assignment 3
- Hand out Assignment 4