Audio Filtrering I MATLAB - Semesterprosjekt

Rapport

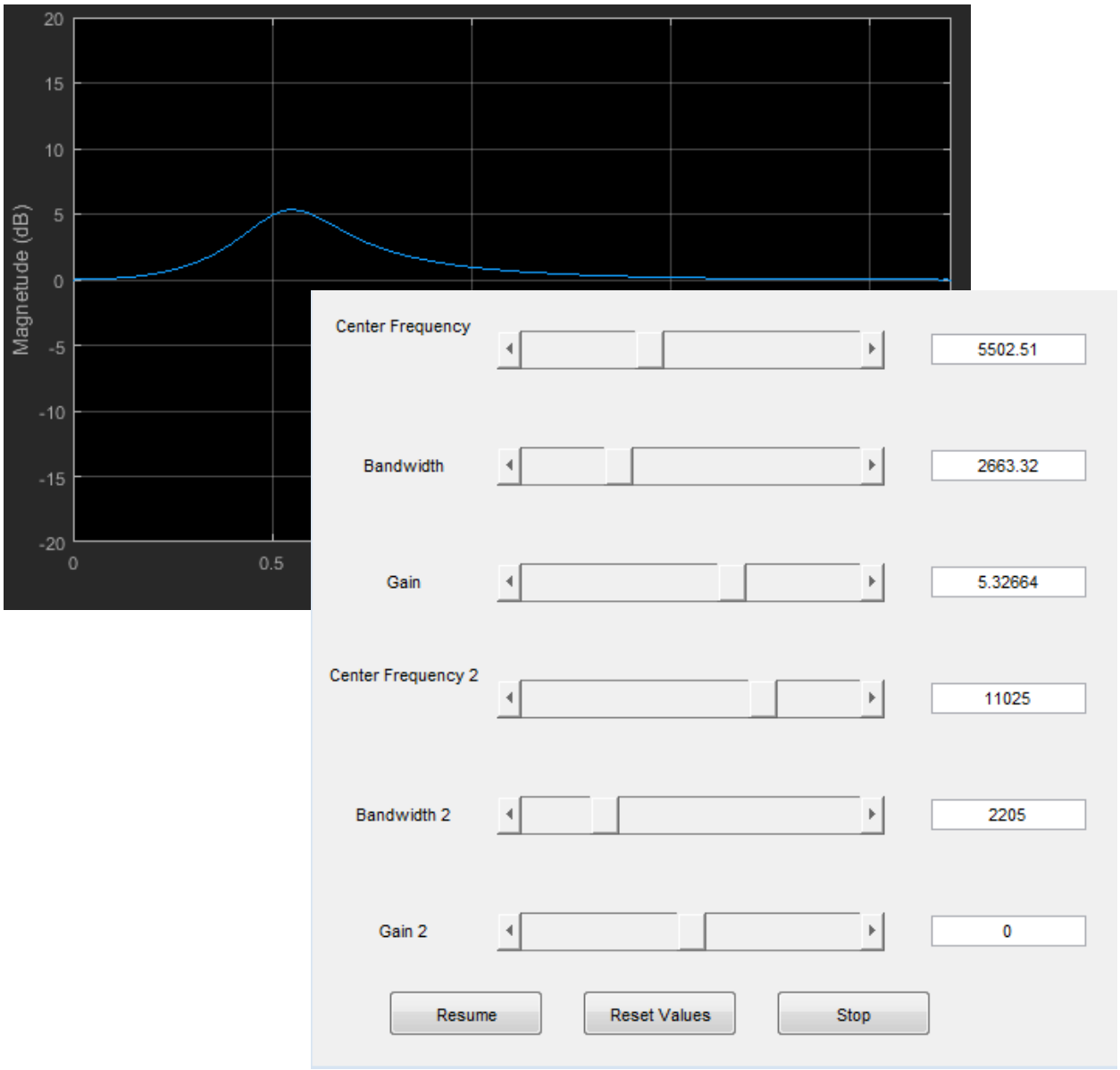
Dynasmiske Systemer - ELTS 2300, Høst 2016

Christophe Plaissy, s300344

Forord

Dette prosjektet går ut på å simulere en equalizer i MATLAB. Det ble valgt denne problemstillingen fordi den er veldig lærerikt angående filtrere generelt (og selvfølgelig Audiobehandling) og muligheter MATLAB tilbyr.

Prosjektet og rapport er utført av Christophe Plaissy, student ved 2. klasse ingeniørfag, Elektronikk og Informasjonsteknologi ved HiOA (HINGELEKTR15H).



Innholdsfortegnelse

Hensikt

Med dette prosjektet er hensikten å forbedre kunnskapen om filtrenes virkemåte, hvordan de er bygget opp og hvilken innvirkning de har for signalbehandling. Vi ønsker å forstå hvordan et filter påvirker lydsignaler, og dermed se hvordan vi kan bruke filtrene til audio-processing.

Beskrivelse

Filtre brukes til veldig mye innenfor Audio-filtering, noise-filtering, data-analytikk og signalbehandling. Størst i dette prosjektet er MATLAB-skriptet, men vi skal ta for oss å se på teori om filtrere og analysere enkelte lav-pass, høy-pass og Butterworth filter. Disse filtrene skal vi kunne bruke i MATLAB simulasjonen gjennom en equalizer.

Teori om filtrere

Ideell filter

Et filter brukes til å filtrere bort (rene opp) ønsket frekvenser i et signal. Et ideelt Lav/høy-pass filter skal kunne filtrere bort med 100% nøyaktighet under/over en bestemt frekvens (j.f. Fig. 1), et ideelt notch-filter skal kunne filtrere bort én bestemt frekvens eller sett av frekvenser, og et ideelt bandpass-filter skal kunne tillate bare én frekvens eller sett av frekvenser. Alle frekvensene som ikke filtreres bort er i Passband området, alle de som filtreres bort er i Stopband området. Et filter er da et verktøy vi kan bruke for signalbehandling – uansett om signalet er strøm, audio, radiobølger …

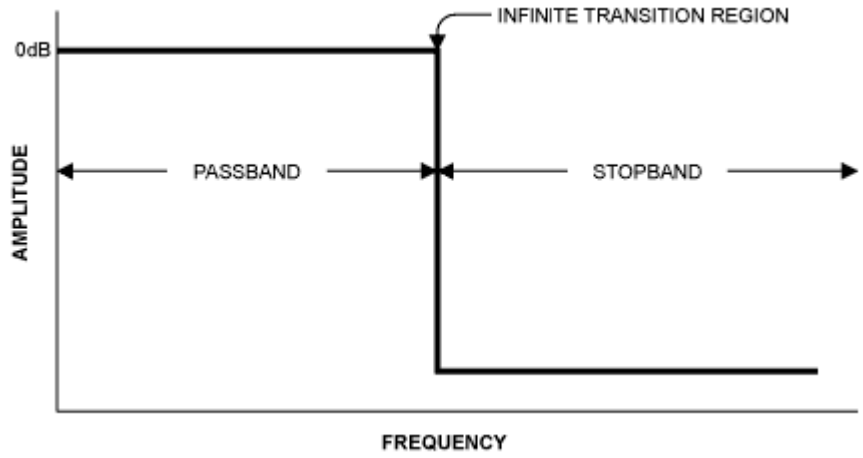


Fig. 1

Ideelle komponenter

Ideelle komponenter oppfører seg som visst i Fig. 2 nedenfor. Impedansen til en motstand er alltid den samme (i gull), impedansen til en kondensator minker når frekvensen øker (i blått) og impedansen til en spole øker når frekvensen øker (i rødt). Selv om ideelle komponenter ikke finnes i virkeligheten forteller dette oss noe om hvordan en krets som inneholder disse komponentene vil oppføre seg, også i forhold til hvordan de er koblet til hverandre.

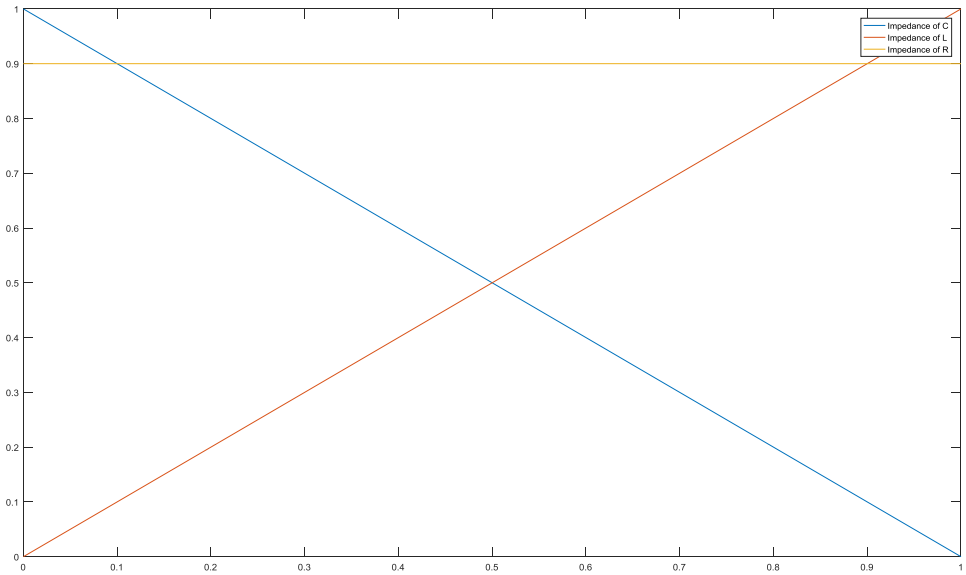


Fig. 2

Center frequency og Bandwidth

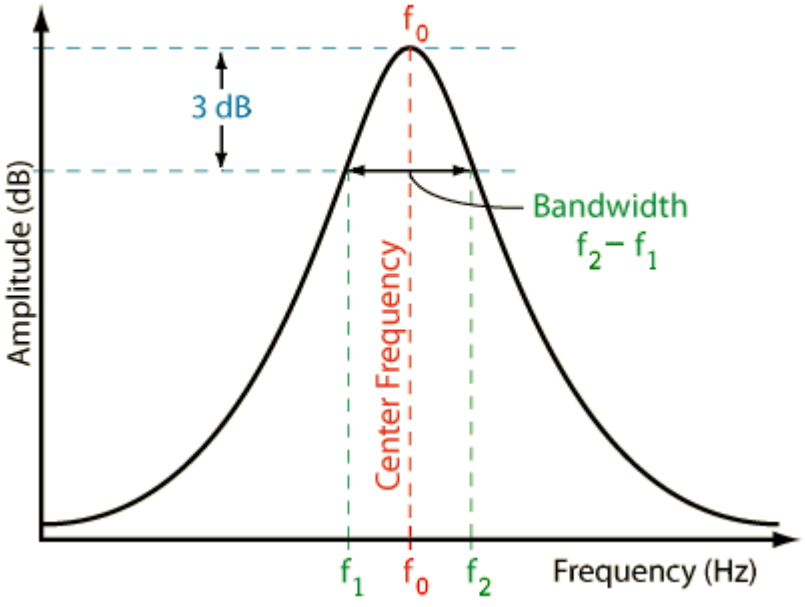


Fig. 3

Cutoff frequency

I figuren over er ''Cutoff frequency'' notert f_1 og f_2 (Fig. 3). Vanligvis skriver vi $f_c = \frac{1}{2\pi\tau}\ [Hz]$ eller $\omega_c = \frac{1}{\tau}\left[\frac{rad}{s}\right]$. Den er frekvensen hvor signalet har 3 dB forskjell fra det originale signalet, eller ønsket amplituden. Denne verdien forteller oss cirka fra hvilke frekvenser vi ønsker å begynne å filtrere.

Forskjellen mellom ulike ordens filtrere

I praksis er det ikke mulig å få en ''Infinite transition region'' som visst i Fig. 1. Ved å bruke et høyere-ordens filter får vi en raskere og bedre (mer nøyaktig) filtrering (Fig. 4):

$$H_1 = \frac{1}{s+1}\ \text{(blått)}$$

$$H_2 = \frac{1}{s^2+s+1}\ \text{(oransje)}$$

$$H_3 = \frac{1}{s^3+2s^2+2s+1}\ \text{(gult)}$$

$$H_4 = \frac{1}{s^4+3s^3+3s^2+3s+1}\ \text{(lilla)}$$

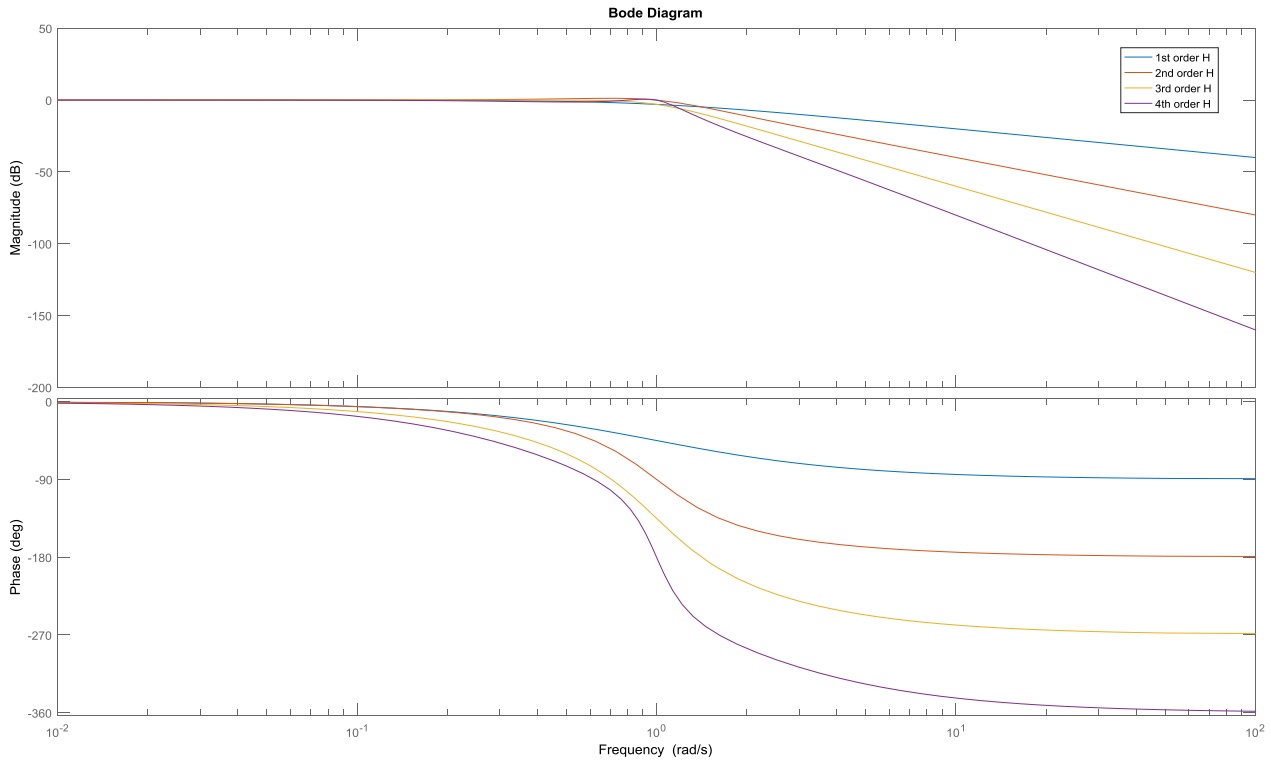


Fig. 4

Modellering

I beregningene som følger er

$$Z_R = R\ [\Omega], \qquad Z_c = \frac{1}{j\omega C}\ [\Omega], \qquad Z_L = j\omega L\ [\Omega], \qquad f = \frac{1}{t}\ [Hz], \qquad \omega = 2\pi f\ \left[\frac{rad}{s}\right],$$
$$\text{og } j = \sqrt{-1} \Leftrightarrow j^2 = -1$$

Vi noterer impedansen til en komponent k som Z_k, f beskriver frekvensen i Hz ($\frac{1}{s}$), ω frekvensen i $\frac{rad}{s}$ og j er et imaginært tall som beskriver vinkelen i kurven til signalet.

Lav-pass filter (Høy-cut/Treble-cut filter)

RC-filter

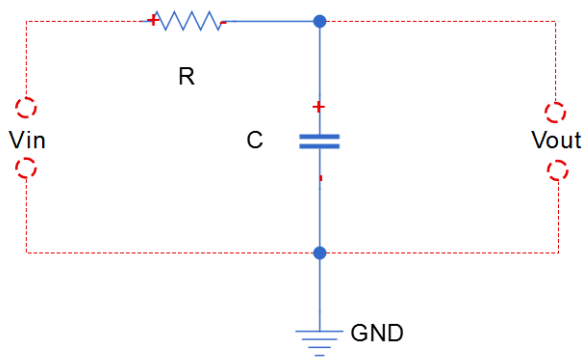


Fig. 5

En annen type lav-pass filter (LC-filter):

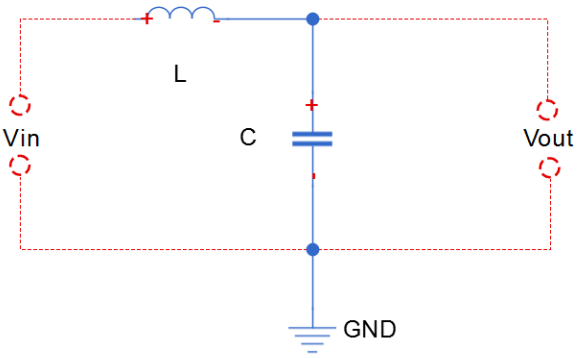


Fig. 6

Vi analyserer det første RC-filteret:

Vi bruker Ohms lov og Kirchoffs Spennings Lov (senere notert KVL):

$$Vi\ har\ at\ V_{in}(s) - iR - iZ_c = 0$$
$$og\ V_{out}(s) = iZ_c = i\frac{1}{sC} \Leftrightarrow i = V_{out}(s)sC$$
$$da\ er\ V_{in}(s) = iZ_c + iR$$
$$\Leftrightarrow V_{in} = V_{out}(s) + V_{out}(s)sRC = V_{out}(s)(sRC + 1)$$
$$\Leftrightarrow V_{out}(s) = \frac{V_{in}(s)}{sRC + 1}$$

Vi setter tidskonstanten $\tau = RC$. Da har vi transferfunksjonen H:

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{1}{s\tau + 1}$$

Da kan vi finne amplituden:

$$|H(\omega)| = \frac{|V_{out}(\omega)|}{|V_{in}(\omega)|} = \frac{1}{\sqrt{(\omega\tau)^2 + 1}}$$

For følgende plot har vi valgt $R = 1000\ \Omega$ og $C = 10e - 6\ F$

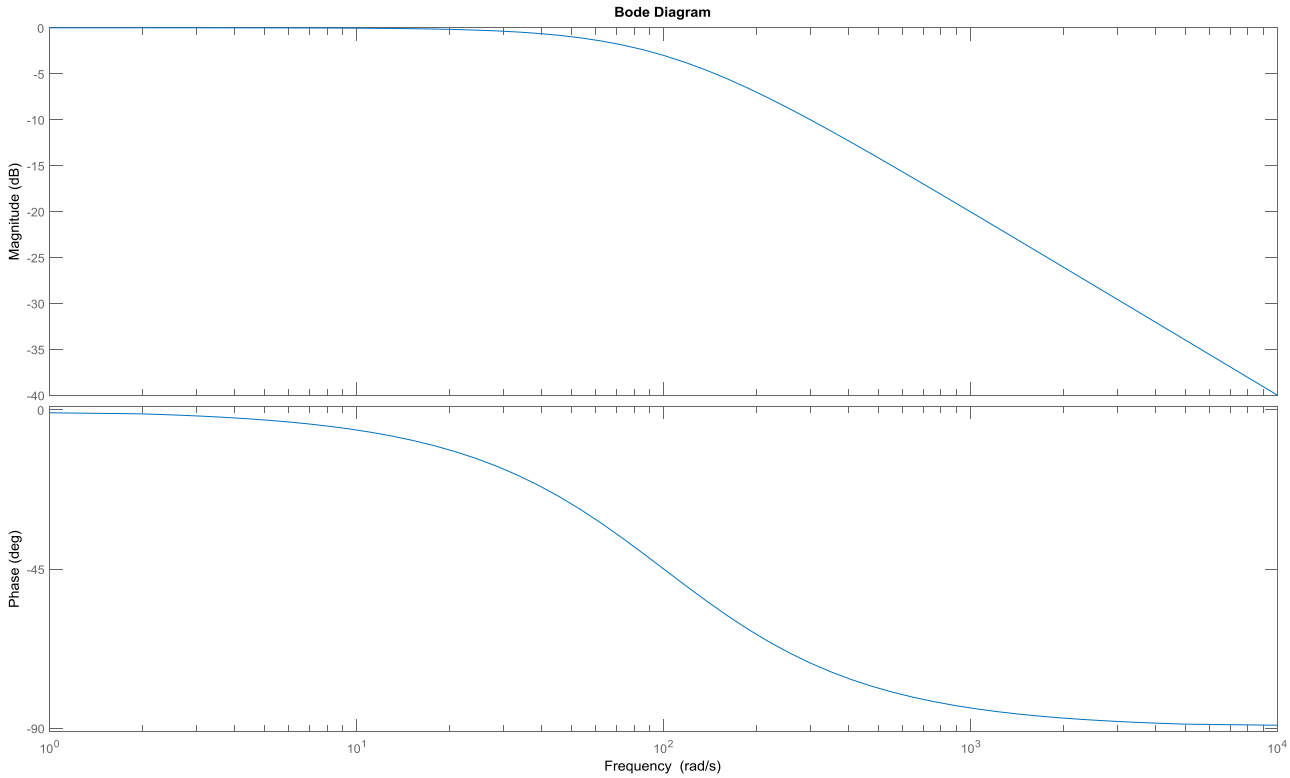


Fig. 7

Høy-pass filter (Lav-cut/Bass-cut filter)

Enkelt CR høy-pass filter:

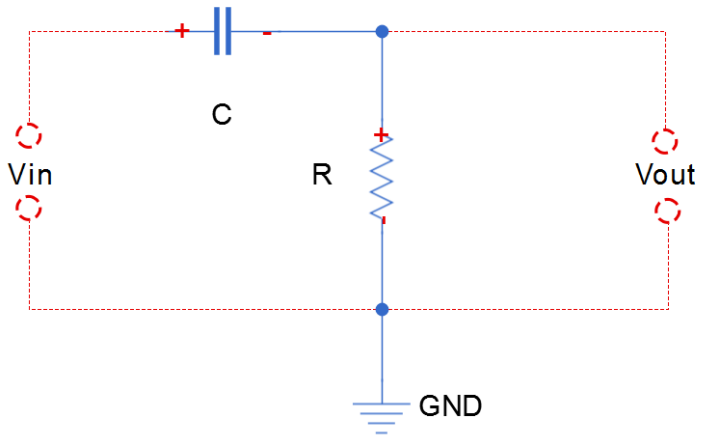


Fig. 8

Vi bruker samme metode med Ohms lov og KVL:

$$Vi\ har\ at\ V_{in}(s) - iR - iZ_c = 0$$

$$V_{in}(s) = iZ_c + iR$$

$$og\ V_{out}(s) = iR \Leftrightarrow i = \frac{V_{out}(s)}{R}$$

$$da\ er\ V_{in}(s) = iZ_c + iR$$

$$\Leftrightarrow V_{in}(s) = \frac{V_{out}(s)}{R}\frac{1}{sC} + \frac{V_{out}(s)}{R}R = V_{out}\left(\frac{1}{sRC} + 1\right)$$

$$\Leftrightarrow V_{out}(s) = \frac{V_{in}(s)}{1 + \frac{1}{sRC}} = V_{in}(s)\left(\frac{sRC}{sRC + 1}\right)$$

Igjen setter vi tidskonstanten $\tau = RC$. Da har vi transferfunksjonen H:

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{s\tau}{1 + s\tau}$$

Da kan vi finne amplituden:

$$|H(\omega)| = \frac{|V_{out}(\omega)|}{|V_{in}(\omega)|} = \frac{\sqrt{(\omega\tau)^2}}{\sqrt{1 + (\omega\tau)^2}} = \frac{\omega\tau}{\sqrt{1 + (\omega\tau)^2}}$$

For følgende plot har vi valgt samme *R* og *C* verdier.

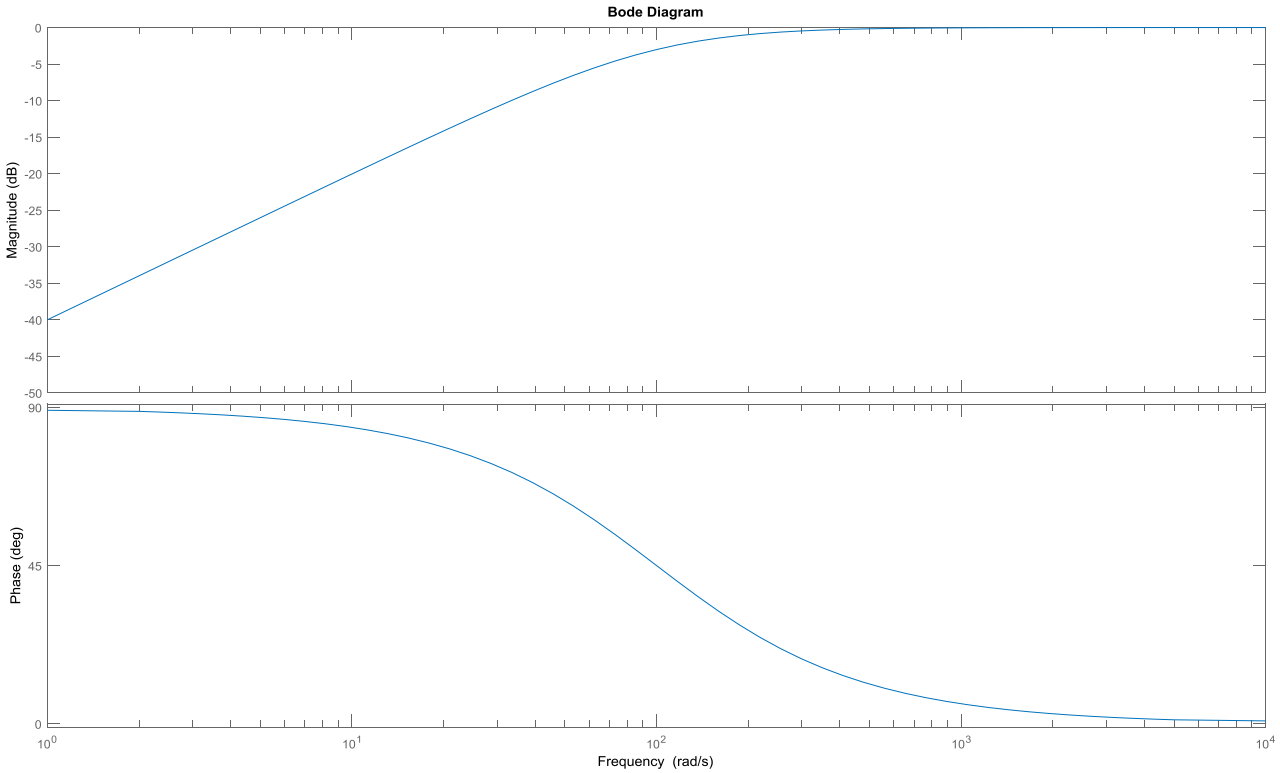


Fig. 9

Band-pass filtere er Lav-pass filter og høy-pass filter koblet i serie og Band-stop filtere (notch) er Lav-pass filter og høy-pass filter koblet i parallell

Butterworth lav-pass filter (3. order)

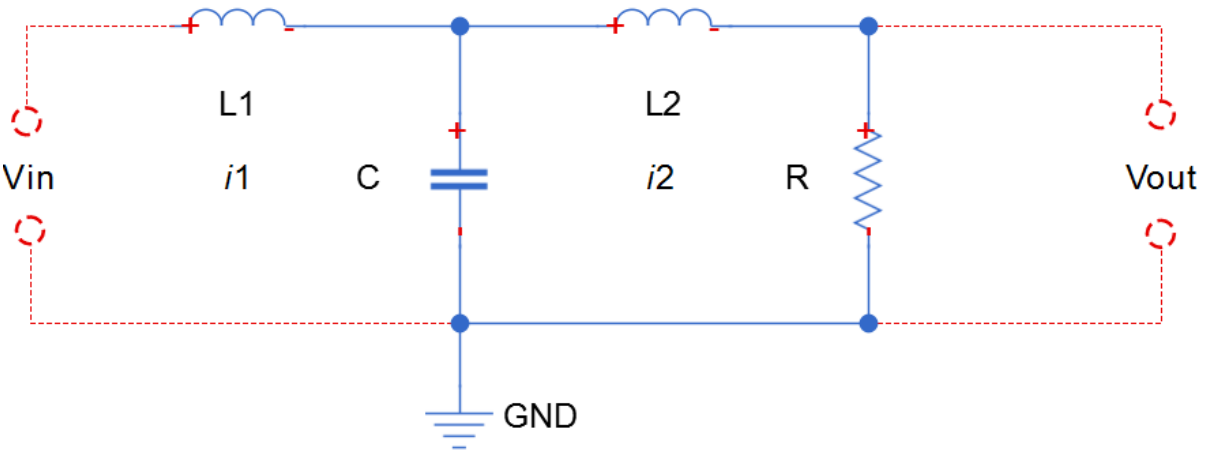


Fig. 10

Ohms lov, KVL og Kierchoffs Strøm Lov (KCL):

$$\left.\begin{array}{l}(1)\quad V_{in}-i_1Z_{L_1}-i_1Z_C+i_2Z_C=0\\(2)\;-i_1Z_c+i_2Z_c-i_2Z_{L_2}-i_2R=0\end{array}\right\}\; i_2R=V_{out},\qquad i_2=\frac{V_{out}}{R}$$

$$\left.\begin{array}{l}(1)\qquad V_{in}-i_1Z_{L_1}-i_1Z_C+\frac{V_{out}}{R}Z_C=0\\(2)\;-i_1Z_c+\frac{V_{out}}{R}Z_c-\frac{V_{out}}{R}Z_{L_2}-V_{out}=0\end{array}\right\}$$

$$(2)\; i_1Z_c-\frac{V_{out}}{R}Z_c+\frac{V_{out}}{R}Z_{L_2}+V_{out}=0$$

$$\Leftrightarrow\; i_1Z_c=\frac{V_{out}}{R}Z_c-\frac{V_{out}}{R}Z_{L_2}-V_{out}$$

$$\Leftrightarrow\; i_1=\frac{V_{out}}{R}-\frac{V_{out}}{RZ_c}Z_{L_2}-\frac{V_{out}}{Z_C}=V_{out}\left(\frac{1}{R}-\frac{Z_{L_2}}{RZ_C}-\frac{1}{Z_C}\right)$$

$$(1)\; V_{in}-V_{out}Z_{L_1}\left(\frac{1}{R}-\frac{Z_{L_2}}{RZ_C}-\frac{1}{Z_C}\right)-V_{out}Z_C\left(\frac{1}{R}-\frac{Z_{L_2}}{RZ_C}-\frac{1}{Z_C}\right)+\frac{V_{out}}{R}Z_C=0$$

$$\Leftrightarrow\; V_{out}\left[Z_{L_1}\left(\frac{1}{R}-\frac{Z_{L_2}}{RZ_C}-\frac{1}{Z_C}\right)-Z_C\left(\frac{1}{R}-\frac{Z_{L_2}}{RZ_C}-\frac{1}{Z_C}\right)+\frac{Z_C}{R}\right]=V_{in}$$

$$\Leftrightarrow\; V_{out}\left[\left(\frac{Z_{L_1}}{R}-\frac{Z_{L_1}Z_{L_2}}{RZ_C}-\frac{Z_{L_1}}{Z_C}\right)-\left(\frac{Z_C}{R}-\frac{Z_{L_2}Z_C}{RZ_C}-\frac{Z_C}{Z_C}\right)+\frac{Z_C}{R}\right]=V_{in},\qquad \text{insetting med } s=j\omega$$

$$\Leftrightarrow\; V_{out}\left[\left(\frac{sL_1}{R}-\frac{sL_1sL_2sC}{R}-sL_1sC\right)-\left(\frac{1}{RsC}-\frac{sL_2}{R}-1\right)+\frac{1}{RsC}\right]=V_{in}$$

$$\Leftrightarrow\; V_{out}\left[\frac{sL_1}{R}+\frac{s^3L_1L_2C}{R}+s^2L_1C-\frac{1}{RsC}+\frac{sL_2}{R}+1+\frac{1}{RsC}\right]=V_{in}$$

$$\Leftrightarrow V_{out} = \frac{R}{R}\frac{V_{in}}{\frac{sL_1}{R} + \frac{sL_1L_2C}{R} + s^2L_1C + \frac{sL_2}{R} + 1}$$

$$\Leftrightarrow V_{out} = \frac{RV_{in}}{sL_1 + s^3L_1L_2C + s^2L_1RC - \frac{1}{sC} + sL_2 + R + \frac{1}{sC}}$$

$$\Leftrightarrow V_{out} = \frac{RV_{in}}{s^3L_1L_2C + s^2L_1RC + sL_1 + sL_2 + R}$$

$$\Leftrightarrow V_{out} = \frac{RV_{in}}{sL_1L_2C + sL_1RC + s(L_1 + L_2) + R}$$

Vi får da tredje ordens transfer funksjonen:

$$H(s) = \frac{V_{out}(s)}{V_{in}(s)} = \frac{R}{s^3L_1L_2C + s^2L_1RC + s(L_1 + L_2) + R}$$

Og kan vi finne amplituden:

$$|H(\omega)| = \frac{|V_{out}(\omega)|}{|V_{in}(\omega)|} = \frac{R}{\sqrt{(\omega^3L_1L_2C)^2 + (\omega^2L_1RC)^2 + \left(\omega(L_1 + L_2)\right)^2 + R^2}}$$

For følgende plot har vi valgt samme $R = 1\ \Omega, C = \frac{4}{3}\ F, L_1 = \frac{3}{2}\ H\ og\ L_2 = \frac{1}{2}\ H$ for å få en enkel

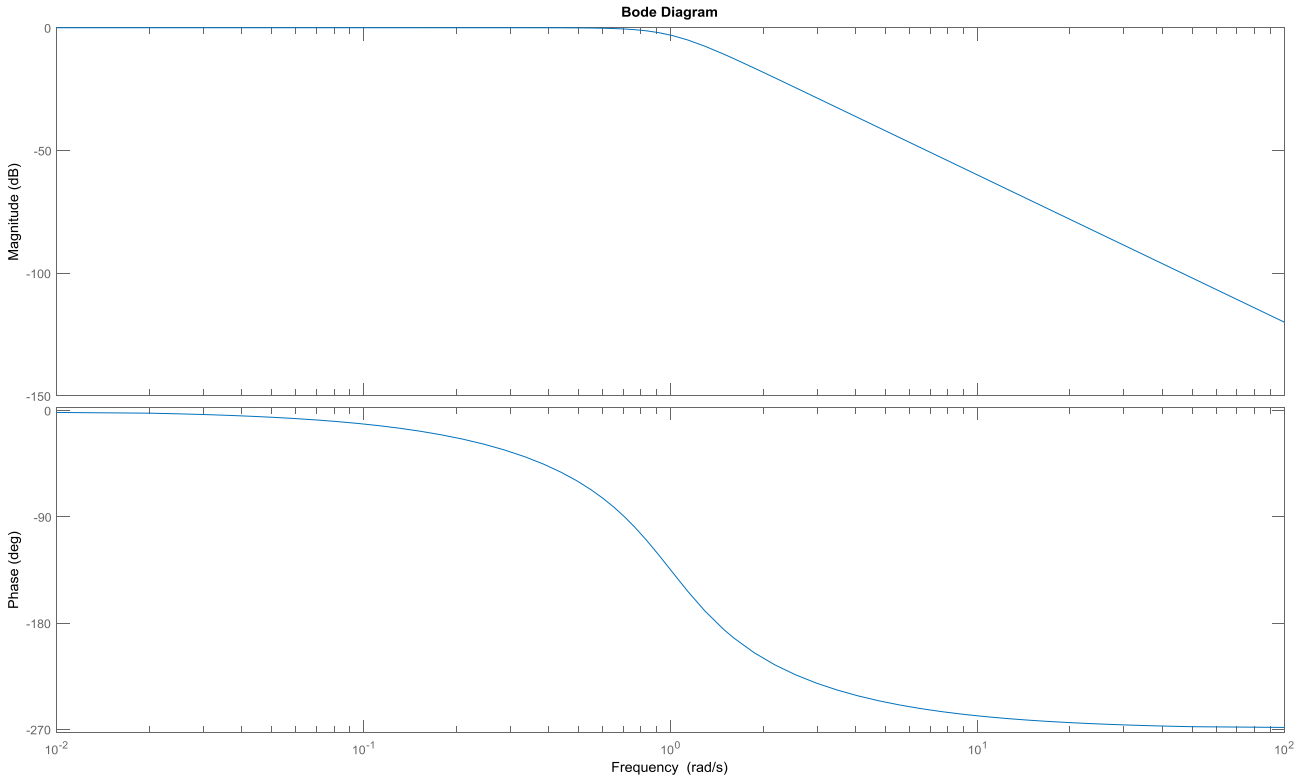


Fig. 11

cutoff verdi $\omega_c = 1$ ($\tau = RCL_1L_2$):

Vi sammenlikner med RC-filteret vi så på tidligere. For begge filtrene har vi valgt verdier av $\omega_c = 1$. Vi ser at Butterworth-filteret (oransje) filtrer bedre og raskere.

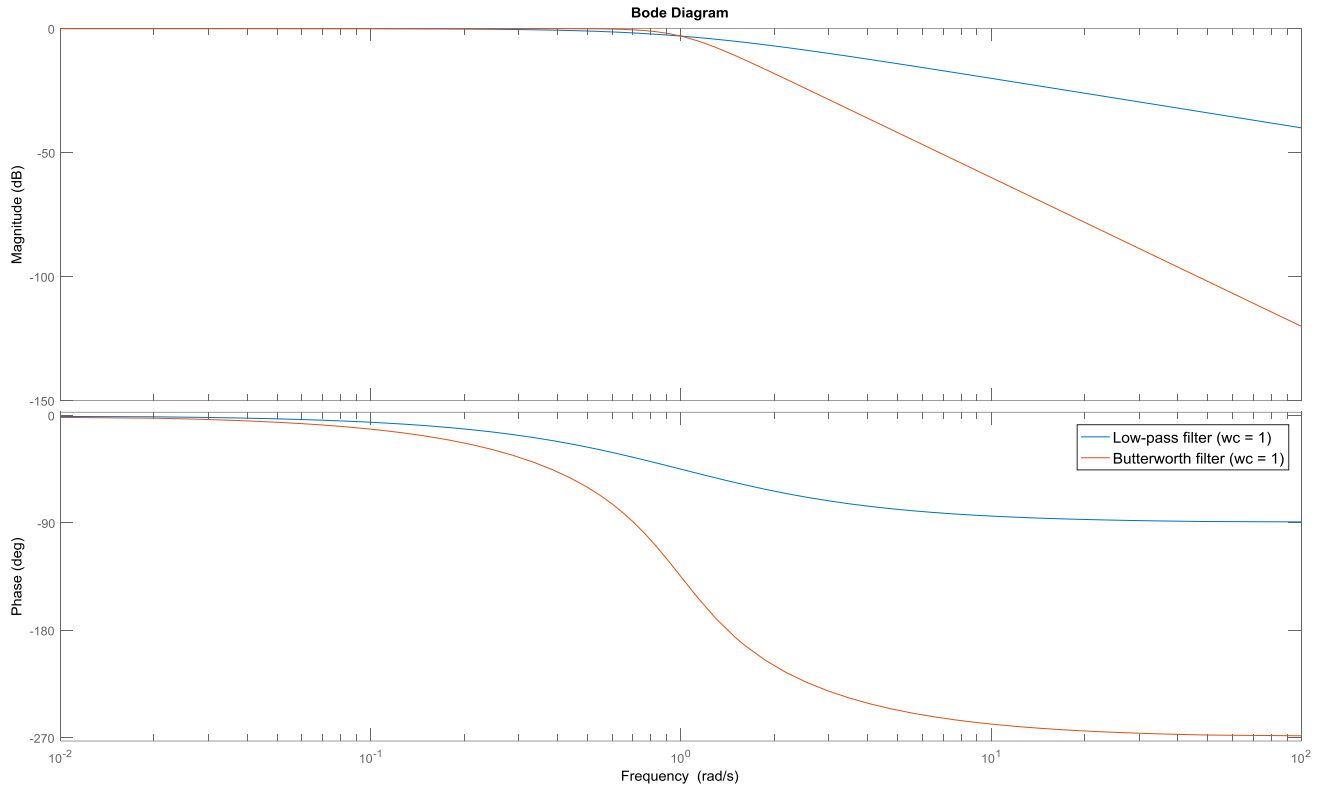


Fig. 12

Audioprosessing i MATLAB

OBS! For å kjøre programmet er det vesentlig å ha dsp-bilbioteket installert på datamaskinen.

Dette programmet lar oss velge to ''Center frequency'', to ''Bandwidth'' og to ''Gain''. Med disse kan vi lage enkle eller kompliserte filtrere. De blir da mulig å simulere to filtrere i serie. Ved å sette enten en av gain'ene eller en av bandwidth'ene til 0, kan vi simulere en høy- eller lav-pass filter.

For å simulere et lav-pass filter (som visst i side 6) med transfer funskjon

$$H(s) = \frac{K}{s\tau + 1}$$

kan vi sette inn disse parametrene: Gain = K, Center Frequency = 0, Bandwidth = ω_C, Gain 2 = 0.

I **Fig. 13 & 14** ser vi denne tilstanden.

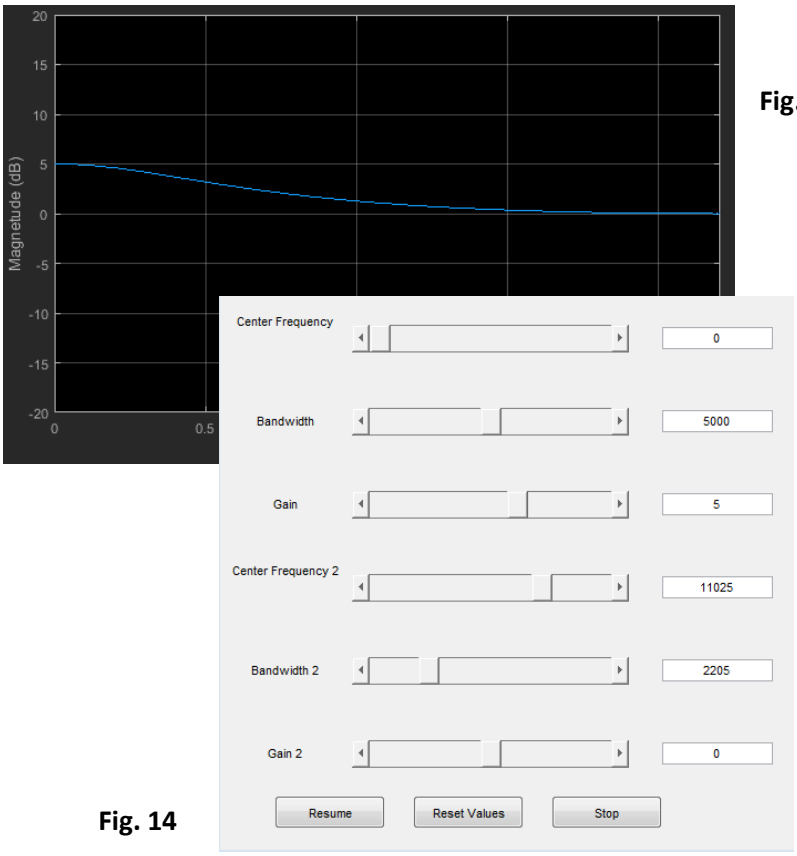


Fig. 13

Fig. 14

På samme måte er det mulig å prøve seg med vanskeligere funksjoner:

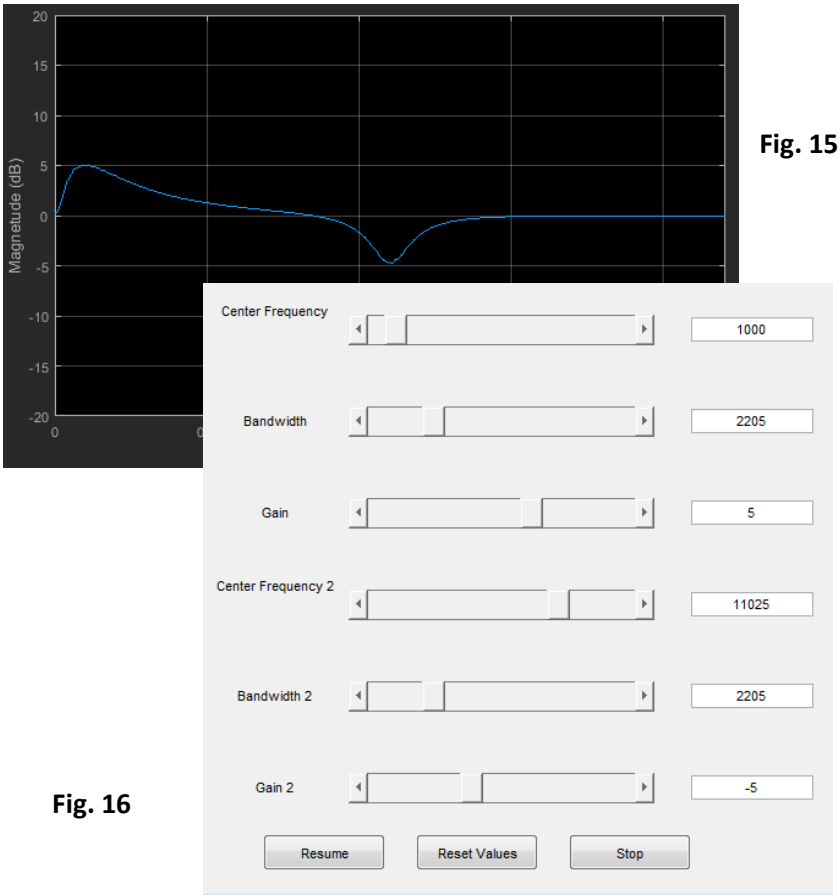


Fig. 15

Fig. 16

Konklusjon

Til konklusjon har filtrere en stor praktisk innvirkning i signalbehandling. MATLAB veldig kraftig til å håndtere signaler på mange forskjellige måter, og det programmet er bare et eksempel på hvordan man kan gjøre dette.

MATLAB skript

Initializer.m

Denne filen starter programmet og kaller alle andre filer. Dersom brukeren ønsker å lese en annen musikk er det da bare å kommentere bort make_music.m fra Initializer.m og erstatte argumentet i dsp.AudioFileReader('music.wav',…) med en annen .wav, .mp3 (osv) fil.

```
% Just to be sure
close all

% Calls the function to make the audio file we want to use
make_music;

% Initiate important variables - We use dsp library from there
SamplesPerFrame = 1024;
audioFile = dsp.AudioFileReader('music.wav','SamplesPerFrame',...
SamplesPerFrame);
fs = audioFile.SampleRate;

% Initiate transfer function
transFunc = dsp.TransferFunctionEstimator('FrequencyRange','onesided',...
                                          'SpectralAverages', 50);
% Initiate plots
arrayPlot = dsp.ArrayPlot('PlotType','Line','YLimits', [-20 20],...
'SampleIncrement',fs/(2*512),...
'YLabel','Magnetude (dB)','XLabel','Frequency (Hz)',...
'Name', 'H=Y/X Transfer Function Estimate',...
'Position',[900,500,600,400]);

spectre = dsp.SpectrumAnalyzer('SampleRate', fs, 'PlotAsTwoSidedSpectrum',
false,...
'SpectralAverages', 20, 'FrequencyScale','Log','Name',...
'AudioOut Spectrum','Position',[1000,50,600,400]);

scope = dsp.TimeScope('SampleRate', fs, 'BufferLength', 4*fs,...
'TimeSpan', 1,'YLimits',[-4 4],'ShowGrid', true,'Name','AudioOut Scope',...
'Position',[400,50,600,400]);

% Initiate player
player = dsp.AudioPlayer('SampleRate', fs);


% Initiate equalizer
GUIequalizer
hpause = findobj('Tag', 'Value');
hstop = findobj('Tag', 'Value');

% tic; % time initialisation, unneeded
while ~isDone(audioFile)
    pauseSim = getappdata(hpause, 'Value');
    stopSim = getappdata(hstop, 'Value');
    if pauseSim == 0
        audioIn = step(audioFile);
    end
    if pauseSim == 1
        drawnow;
```

```
        continue;
    end

    if stopSim == 1
        break;
    end

    audioOut = audioAlgorithm(audioIn); % process audio Out
    H = step(transFunc, audioIn, audioOut); % transfer function
    step(arrayPlot, 20*log10(abs(H)));  % plots result as arrayplot

    % if necessary, to analyse audioOut from other perspectives
    step(spectre, audioOut); % plots result as spectre
    step(scope, audioOut); % plots result as scope

    step(player, audioOut); % plays result % comment out to suppress sound
end

close 'Tuning GUI'

% Release the dsp components for a cleaner quit
release(arrayPlot)
release(scope)
release(spectre)
release(player)
release(transFunc)
```

make_music.m

Denne filen brukes for å vise hva en Audiofil ser ut som. En Audiofil er en matrise med verdier og det er mulig å endre på den for å øke gain for eksempel eller mye annet.

```
% we want to make one file out of the free .wav we have
function make_music
    fs = 44100;
    %frame_duration = .1; % duration of a frame, unneeded here
    %frame_length = frame_duration*fs; % length of a frame, unneeded here

    % Create the file if it doesn't exist
    if exist('music.wav','file') ~= 2

        bass = audioread('bass.wav');
        drums = audioread('drums.wav');
        guitar = audioread('guitar.wav');

        Nb = length(bass);
        Nd = length(drums);
        Ng = length(guitar);

        % Find the longest file
        if Nb >= Nd
            Nmax = Nb;
        else
            Nmax = Nd;
        end
        if Ng >= Nmax
            Nmax = Ng;
        end

        % Give all files same length
        bass(Nb:Nmax)=0;
        drums(Nd:Nmax)=0;
        guitar(Ng:Nmax)=0;

        % we make music.wav with two channels, and will probably try later
on to influence one of them at a time
        music = zeros(Nmax,2);
        music(:,1) = bass + drums + guitar;
        music(:,2) = bass + drums + guitar;

        % Create the final file
        audiowrite('music.wav',music,fs);

    % If the file already exists
    else
        % then do nothing
    end
end
```

GUIequalizer.m

Denne filen lager den Graphic User Interface for Equlizer'en.

```
%% GUIequalizer - main function
% In this script, we set 'Tag' values to the different graphic buttons to
% be able to find and use them in other functions
function GUIequalizer
    % Create the Graphic User Interface (GUI) for the equaliser
    figure('Name','Tuning GUI','Position',[100,400,520,500],...
'Resize','off');

    % Construct the components, their Tags, their values and connexions
    % frequency1
    uicontrol('Style','text','String','Center Frequency','Position',...
[10, 450, 100, 35],'Tag','frequencytitle1');

    uicontrol('Style','slider','Min',0,'Max',15000,'Value',11025,...
'Position',[120,450,250,25],Tag','frequency1',...
'Callback',{@frequency1_callback});
    uicontrol('Style','edit', 'String', '11025',...
'Position',[400,453,100,20],'Tag','frequencycontrol1', ...
'Callback',{@frequencycontrol1_callback});

    % bandwidth1
    uicontrol('Style','text','String','Bandwidth',...
'Position',[10,375,100,20],'Tag','bandwidthtitle1');
    uicontrol('Style','slider','Min',0,'Max',10000,'Value',2205,...
'Position',[120,375,250,25],'Tag','bandwidth1',...
'Callback',{@bandwidth1_callback});
    uicontrol('Style','edit', 'String','2205',...
'Position',[400,378,100,20],'Tag','bandwidthcontrol1',...
'Callback',{@bandwidthcontrol1_callback});

    % gain1
    uicontrol('Style','text','String','Gain',...
'Position',[10, 300, 100, 20],'Tag','gaintitle1');
    uicontrol('Style','slider','Max',20,'Min',-20,'Value',1, ...
'Position',[120,300,250,25],'Tag','gain1','Callback',{@gain1_callback});
    uicontrol('Style','edit','String','1','Position',[400,303,100,20],...
'Tag','gaincontrol1','Callback',{@gaincontrol1_callback});

    % frequency2
    uicontrol('Style','text','String','Center Frequency 2',...
'Position',[10,225,100,35],'Tag','frequencytitle2');
uicontrol('Style','slider','Min',0,'Max',15000,'Value',11025,...
'Position',[120,225,250,25],'Tag','frequency2',...
'Callback',{@frequency2_callback});
    uicontrol('Style','edit', 'String', '11025',...
'Position',[400,228,100,20],'Tag', 'frequencycontrol2',...
'Callback',{@frequencycontrol2_callback});

    % bandwidth2
    uicontrol('Style','text','String','Bandwidth 2',...
'Position',[10,150,100,20],'Tag','bandwidthtitle2');
    uicontrol('Style','slider','Min',0,'Max',10000,'Value',2205,...
'Position',[120,150,250,25],'Tag','bandwidth2',...
'Callback',{@bandwidth2_callback});
    uicontrol('Style','edit','String','2205',...
'Position',[400,153,100,20],'Tag',...
```

```
'bandwidthcontrol2','Callback',{@bandwidthcontrol2_callback});

    % gain2
    uicontrol('Style','text','String','Gain 2',...
'Position',[10, 75, 100, 20],'Tag','gaintitle2');
    uicontrol('Style','slider','Max',20,'Min',-20,'Value',0,...
'Position',[120,75,250,25],'Tag','gain2','Callback',{@gain2_callback});
    uicontrol('Style','edit', 'String', '0',...
'Position',[400,78,100,20],'Tag','gaincontrol2',...
'Callback',{@gaincontrol2_callback});

    % reset
    uicontrol('Style','pushbutton','Callback',{@resetButton_callback},...
'String','Reset Values',...
'Position',[175,20,100,30],'Tag','treset');

    % pause
    guiPause = uicontrol('Style','pushbutton',...
'Callback',{@pauseButton_callback},'String','Pause',...
'Position',[50,20,100,30], 'Tag', 'tpause');
    % stop
    guiStop = uicontrol('Style','pushbutton',...
'Callback',{@stopButton_callback},'String','Stop',...
'Position', [300,20,100,30], 'Tag', 'tstop');
    % storing data for pause and stop with setappdata for later use on
other functions
    setappdata(guiPause, 'Value', 0);
    setappdata(guiStop, 'Value', 0);
end

%% function pause
    function  pauseButton_callback(~, ~)
        hpause = findobj('Tag','tpause');
        paused = getappdata(hpause, 'Value');
        switch paused
            case 1
                setappdata(hpause, 'Value', 0);
                set(hpause,'String','Pause');
            case 0
                setappdata(hpause, 'Value', 1);
                set(hpause,'String','Resume');
        end
    end

%% function to stop/quit
    function stopButton_callback(~, ~)
        hstop = findobj('Tag', 'tstop');
        setappdata(hstop,'Value',1);
    end

%% function to reset all values
function resetButton_callback(~, ~)
    % frequency1 and 2
    hfrequency1 = findobj('Tag','frequency1');
    set(hfrequency1,'Value',11025);
    frequency1_callback;
    hfrequency2 = findobj('Tag','frequency2');
    set(hfrequency2,'Value',11025);
    frequency2_callback;
```

```
    % bandwidth1 and 2
    hbandwidth1 = findobj('Tag','bandwidth1');
    set(hbandwidth1,'Value',2205);
    bandwidth1_callback;
    hbandwidth2 = findobj('Tag','bandwidth2');
    set(hbandwidth2,'Value',2205);
    bandwidth2_callback;

    % gain1 and 2
    hgain1 = findobj('Tag','gain1');
    set(hgain1,'Value',1);
    gain1_callback;
    hgain2 = findobj('Tag','gain2');
    set(hgain2,'Value',0);
    gain2_callback;
end

%% functions to modify volumes
%% frequency1 and 2
function frequency1_callback(~, ~)
    hfrequency1 = findobj('Tag','frequency1');
    hfrequencycontrol1 = findobj('Tag','frequencycontrol1');
    frequency1 = get(hfrequency1,'Value');
    set(hfrequencycontrol1,'String',frequency1);
end
function frequencycontrol1_callback(~,~)
    hfrequencycontrol1 = findobj('Tag','frequencycontrol1');
    hfrequency1 = findobj('Tag','frequency1');
    frequency1 = get(hfrequencycontrol1,'String');
    frequency1 = str2num(frequency1);
    set(hfrequency1,'Value',frequency1);
end

function frequency2_callback(~, ~)
    hfrequency2 = findobj('Tag','frequency2');
    hfrequencycontrol2 = findobj('Tag','frequencycontrol2');
    frequency2 = get(hfrequency2,'Value');
    set(hfrequencycontrol2,'String',frequency2);
end
function frequencycontrol2_callback(~,~)
    hfrequencycontrol2 = findobj('Tag','frequencycontrol2');
    hfrequency2 = findobj('Tag','frequency2');
    frequency2 = get(hfrequencycontrol2,'String');
    frequency2 = str2num(frequency2);
    set(hfrequency2,'Value',frequency2);
end

%% bandwidth1 and 2
function bandwidth1_callback(~, ~)
    hbandwidth1 = findobj('Tag','bandwidth1');
    hbandwidthcontrol1 = findobj('Tag','bandwidthcontrol1');
    bandwidth1 = get(hbandwidth1,'Value');
    set(hbandwidthcontrol1,'String',bandwidth1);
end
function bandwidthcontrol1_callback(~, ~)
    hbandwidth1 = findobj('Tag','bandwidth1');
    hbandwidthcontrol1 = findobj('Tag','bandwidthcontrol1');
    bandwidth1 = get(hbandwidthcontrol1,'String');
    bandwidth1 = str2num(bandwidth1);
    set(hbandwidth1,'Value',bandwidth1);
end
```

```
function bandwidth2_callback(~, ~)
    hbandwidth2 = findobj('Tag','bandwidth2');
    hbandwidthcontrol2 = findobj('Tag','bandwidthcontrol2');
    bandwidth2 = get(hbandwidth2,'Value');
    set(hbandwidthcontrol2,'String',bandwidth2);
end
function bandwidthcontrol2_callback(~, ~)
    hbandwidth2 = findobj('Tag','bandwidth2');
    hbandwidthcontrol2 = findobj('Tag','bandwidthcontrol2');
    bandwidth2 = get(hbandwidthcontrol2,'String');
    bandwidth2 = str2num(bandwidth2);
    set(hbandwidth2,'Value',bandwidth2);
end

%% gain1 and 2
function gain1_callback(~, ~)
    hgain1 = findobj('Tag','gain1');
    hgaincontrol1 = findobj('Tag','gaincontrol1');
    gain1 = get(hgain1,'Value');
    set(hgaincontrol1,'String',gain1);
end
function gaincontrol1_callback(~, ~)
    hgain1 = findobj('Tag','gain1');
    hgaincontrol1 = findobj('Tag','gaincontrol1');
    gain1 = get(hgaincontrol1,'String');
    gain1 = str2num(gain1);
    set(hgain1,'Value',gain1);
end

function gain2_callback(~, ~)
    hgain2 = findobj('Tag','gain2');
    hgaincontrol2 = findobj('Tag','gaincontrol2');
    gain2 = get(hgain2,'Value');
    set(hgaincontrol2,'String',gain2);
end
function gaincontrol2_callback(~, ~)
    hgain2 = findobj('Tag','gain2');
    hgaincontrol2 = findobj('Tag','gaincontrol2');
    gain2 = get(hgaincontrol2,'String');
    gain2 = str2num(gain2);
    set(hgain2,'Value',gain2);
end
```

audioAlgorithm.m

Denne filen håndterer Audio algoritmen.

```
%% This algorithm takes in an audio and gives out an audio based on
modifications made in the GUIequalizer
function y = audioAlgorithm(u)

    persistent PE1 PE2;
    hfrequency1 = findobj('Tag','frequency1');
    frequency1 = get(hfrequency1,'Value');
    hfrequency2 = findobj('Tag','frequency2');
    frequency2 = get(hfrequency2,'Value');

    hbandwidth1 = findobj('Tag','bandwidth1');
    bandwidth1 = get(hbandwidth1,'Value');
    hbandwidth2 = findobj('Tag','bandwidth2');
    bandwidth2 = get(hbandwidth2,'Value');

    hgain1 = findobj('Tag','gain1');
    gain1 = get(hgain1,'Value');
    hgain2 = findobj('Tag','gain2');
    gain2 = get(hgain2,'Value');

    if isempty(PE1)
        PE1 = dsp.ParametricEQFilter('Bandwidth', bandwidth1,...
'CenterFrequency', frequency1, 'PeakGaindB', gain1);
        PE2 = dsp.ParametricEQFilter('Bandwidth', bandwidth2,...
'CenterFrequency', frequency2, 'PeakGaindB', gain2);
    end

    % u = audio In
    [PE1, PE2] = processParameters(PE1, PE2);
    v = step(PE1, u); % v = audio In after PE1 modifications
    y = step(PE2, v); % y = audio Out after PE1 and PE2 modifications
end

%% GUIequalizer part
function [PE1, PE2] = processParameters(PE1, PE2)

    hfrequency1 = findobj('Tag','frequency1');
    frequency1 = get(hfrequency1,'Value');
    hfrequency2 = findobj('Tag','frequency2');
    frequency2 = get(hfrequency2,'Value');

    hbandwidth1 = findobj('Tag','bandwidth1');
    bandwidth1 = get(hbandwidth1,'Value');
    hbandwidth2 = findobj('Tag','bandwidth2');
    bandwidth2 = get(hbandwidth2,'Value');

    hgain1 = findobj('Tag','gain1');
    gain1 = get(hgain1,'Value');
    hgain2 = findobj('Tag','gain2');
    gain2 = get(hgain2,'Value');

    PE1 = dsp.ParametricEQFilter('Bandwidth',bandwidth1,...
'CenterFrequency',frequency1,'PeakGaindB',gain1);
    PE2 = dsp.ParametricEQFilter('Bandwidth',bandwidth2,...
'CenterFrequency',frequency2,'PeakGaindB',gain2);
end
```