22051 - Assignment number 1

GROUP NUMBER 1, Cecilie Christensen s183527, Kelan Solomon s212098, Jonas Christoffersen s163944, Eline Dorothea Siegumfeldt s183540
Github: https://github.com/kelansolo/SS-assignment-1-.git

I. Summary and objectives

The purpose of this report is to present an understanding of some principles behind discrete time signals and use this understanding to solve, analyse and present relevant problems. The report covers theory within multiple subjects such as domains, the effect of poles and zeros, filters and filter design. Furthermore the theory is applied to hands-on assignments, which are used to gain practical knowledge. What was learned from this whole assignment is analysing and recreating signals and sounds with different tools from DSP, which will be mentioned in the following.

II. Methods

The most important methods used in this report include matlab's function *fft* and *ifft*, which is the discrete fourier transform (DFT) and the inverse dicrete fourier transform (IDFT). The fourier transform transforms a signal from the time domain to the frequency domain and the inverse fourier transform does the opposite: it transforms the signal back to the time domain from the frequency domain. The math behind *ift* and *ifft* can be seen in eq. (1) and eq. (2).

$$X(r\omega_0) = \sum_{n=0}^{N_0-1} x(n)e^{-jr\omega_0 n} \tag{1}$$

$$x(n) = \frac{1}{N_0}\sum_{r=0}^{N_0-1} X(r\omega_0)e^{jr\omega_0 n} \tag{2}$$

Where ω_0 refers to the spectral resolution, $\omega_0 = \frac{2\pi fs}{N_0}$ and N_0 to the number of samples. DFT is used to examine the frequencies a sound signal is composed of. In the piano exercise DFT is for example used to find the fundemental frequency and to determine the number of necessary components N when synthesising the piano signal. By working in the frequency domain is it possible to recreate complex signals, which is almost impossible when working in the time domain. Therefore the synthesised piano signal is created in the frequency domain and then transfered back to the time domain by using matlab's *ifft*. DFT and IDFT are also relevant when calculating the output signal of a system e.g. a filter. It is easier and more efficient to multiply the input signal with a transfer function in the frequency domain and then do an IDFT than it is to convolve the input signal and the impulse response in the time domain. This has been used in matlab's function *filter* in the recursive filter exercise; it takes a system's transfer function and a input signal as input, and returns the filtered output. Matlab's functions *zplane* and *freqz* also have been heavily used in order to design the filters throughout this report. In fig. (1), *zplaine* has been used: it shows where the poles and zeros of the transfer function for a running sum filter of order 3 and 5 are in the z-plane. The crosses show the poles and the rings show the zeros. From these poles and zeros in the z-plane, it is possible to determine the transfer function and frequency response of an arbitrary system. The transfer function of a system can be described as:

$$H(z) = \frac{(z-\zeta_1)(z-\zeta_2)...(z-\zeta_n)}{(z-\gamma_1)(z-\gamma_2)...(z-\gamma_n)} \tag{3}$$

where ζ_n are the zeros of the system and γ_n are the poles of the system. By using matlab's function *freqz* it is possible to plot the frequency response of a system as it can be seen in fig. (2) and (3).

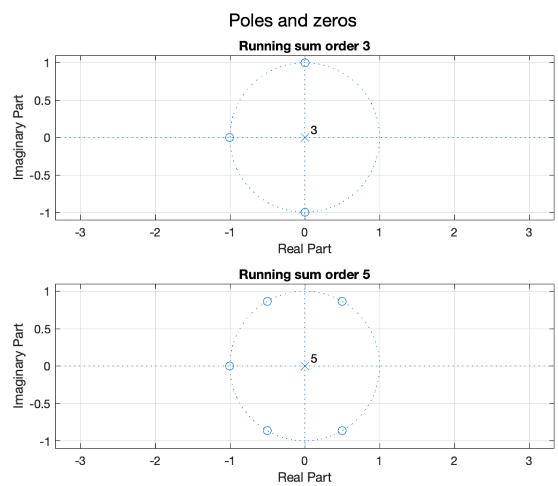

Fig. 1. Poles and zeros plotted in the z-plane for a running sum filter order 3 and 5

By examining the frequency response we can determine which frequencies the system attenuates and amplifies. In fig. (5) and (4) it is, for example, clear the designed filter blocks out the normalised frequency 0.1.

In the recursive filter exercise we are asked to design a FIR (finite impulse response) and an IIR (infinite impulse response) filter. The difference between these two filters is the coefficients in the transfer function $H(z)$. The transfer function can also be described as:

$$\frac{Y(z)}{X(z)} = H(z) = \frac{b_0 z^n + b_1 z^{n-1} + ... + b_n z^0}{a_0 z^m + a_1 z^{m-1} + ... + a_m z^0} \tag{4}$$

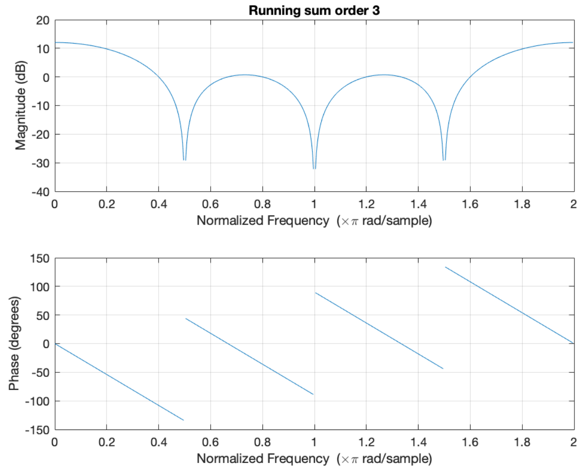

Fig. 2. Frequency response of the running sum filter order 3

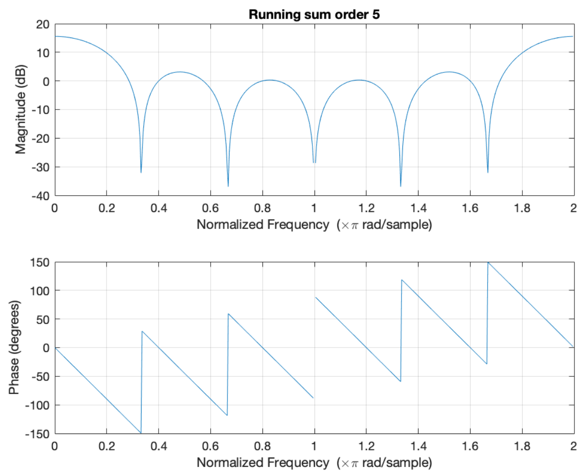

Fig. 3. Frequency response of the running sum filter order 5

b_n are the coefficients for the delayed input or feed-forward coefficients and a_n are the coefficients for the delayed output or feed-back coefficients. The delayed input contributes to the current output when there are b_n coefficients and the delayed output contributes to the current output when there are a_n coefficients. The IIR filter therefore has a_n coefficients because the output is fed back making the output of the system infinite in duration, whereas the FIR filter only has b_n coefficients. The IIR filter can have b_n coefficients too.

Another important element to consider when designing filters is aliasing. In order to avoid aliasing, the sampling frequency f_s should be twice the nyquist frequency, f_n, so $f_s = 2f_n$: if a signal has a frequency of 100 Hz, the minimum sampling frequency should be 200 Hz. This means the nyquist frequency becomes 100 Hz. Normally the nyquist frequency is set to be a bit larger than the largest frequency sent through the system.

To recreate the sound from a piano, we first transform the time signal into the frequency domain, we can observe that the

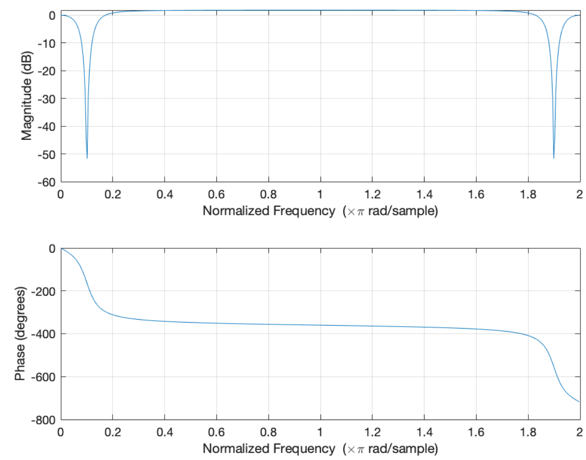

Fig. 4. Frequency response of the filter that blocks the normalised frequency of 0.1. The poles are placed at the same angle as the zero, at the distance $R = \frac{9}{10}$ from the center of the z-plane

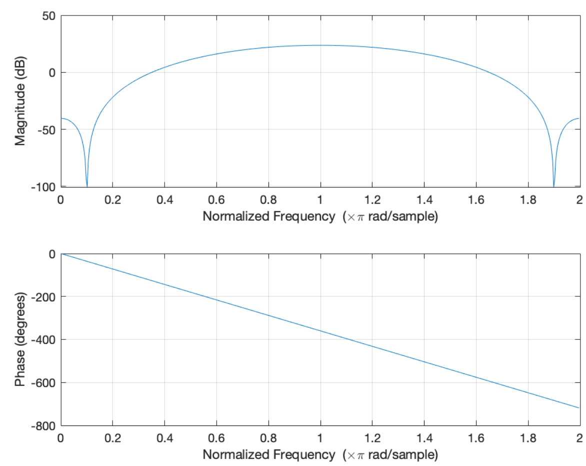

Fig. 5. Frequency response of the filter that blocks the normalised frequency of 0.1. The poles are placed at the center of the z-plane

fundamental frequency is around 130 Hz. As the signal is a sine with no initial phase, we know that the Fourier transform will be strictly imaginary. A vector of zeros is created and -0.5i is then added to the spot corresponding to the positive frequency and 0.5i is added to the corresponding negative frequency location. For example for a single harmonic, this vector would be of the form

$$[0, 0, .., 0, -0.5, 0, ..., 0, 0.5, 0, ..., 0, 0]$$

Where the first half of the vector would be the positive frequencies and the second half the negative (figure 6).

The coefficients(± 0.5) can be found by developing Euler's formula resulting in:

$$sin(\omega t) = \frac{-e^{i\omega t}}{2} i + \frac{e^{-i\omega t}}{2} i$$

This can also been seen as the energy being split over the positive and negative frequencies. This vector is then passed

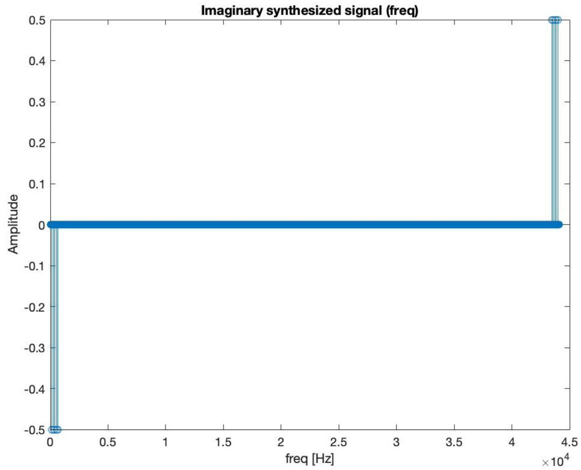

Fig. 6. Imaginary frequency plot of a synthesised piano with 5 harmonics

into the time domain and compared to the original signal. We next added a phase shift to our signal, as seen in figure 7 by developing $sin(\omega t + \phi)$ using Euler's formula (ϕ being the phase shift from the original signal).

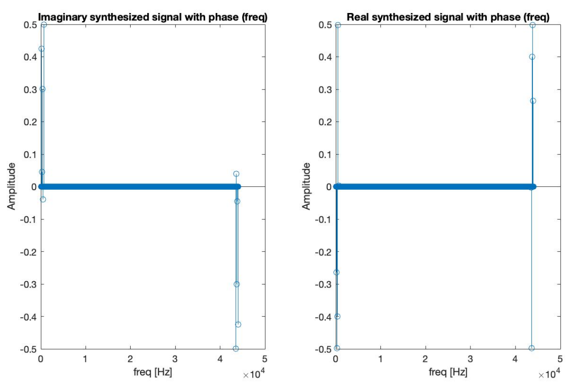

Fig. 7. Imaginary and real frequency plots of a synthesised piano with 5 harmonics and with phase shift

III. Results

A. 1.1

The poles and zeros of the two filters of orders $N = 3$ and $N = 5$ have been found through a z-transform and are listed in the table (I). The poles and zeros have also been

Poles H1		0		0	0
Zeros H1		i		1	$-i$
Poles H2	0	0	0	0	0
Zeros H2		$0.5 \pm 0.866i$		$-0.5 \pm 0.866i$	-1

TABLE I
Poles and zeros of H1 and H2

plotted, and are shown in figure 1. It is known that zeros attenuate frequencies while poles amplify them, meaning that when two transfer functions are inverted, the gain becomes positive instead of negative and as a result the phase shift changes direction.

From the frequency response magnitude plots, figure 2 and figure 3, it can be seen that every zero makes a steep attenuation and the phase jumps 180 degrees at every zero. For the third order running sum we see attenuation at $\omega = \frac{\pi}{2}$, $\omega = \pi$ and $\omega = \frac{3\pi}{2}$, which corresponds to the angle to each zero in the z-plane in figure (1). The frequency response of the filter which blocks normalised frequencies of 0.1 is seen in figure (5). It is obvious that the normalised frequency of 0.1 is attenuated compared to the other frequencies, which is a result of the four zeros placed in the z-plane. When adding four poles on a circle of radius $\frac{9}{10}$ to the same filter, we can see, from the magnitude plot, that the attenuation is more concentrated and that the phase becomes more steep around 0.1. This means that when the poles are approaching the zeros at the same angle as the zeros, the attenuation and the phase shift become more concentrated around the normalised frequency 0.1.

B. 1.2

When analysing the different filters which were created, the results are as expected: the low-pass attenuates high frequency and the high-pass low frequency sinuses. The sine is the same after going through the all-pass filter. Some spikes in the phase graph may indicate noise. The high pass filter cuts out the bass in *mini-me.wav*, the low pass on the other hand makes the sound seem muffled and the all pass doesn't make a noticeable difference. Changing the sampling frequency will change the Nyquist frequency thus changing the the suppressed frequencies. If the the sampling frequency is raised then the cutoff frequency is higher as more frequencies are "squeezed" onto the semicircle of the pole-zero plot thus more frequencies will be suppressed. More poles and zeros, if placed correctly, can result in a steeper attenuation after the cut off frequency. The pole-zero plot can be " unwrapped" from the unity circle to visualise what the transfer function would reassemble

C. 2.1

The echos created from the delayed impulses with IIR and FIR filters have quite a similar sound. When listening to the sounds with a 10ms delay we can hear that the sound has changed a bit, but it is not obvious that an echo has been added. For the delay of 200ms it is very easy to determine that an echo has been added, since everything is being heard twice. The FIR filter sounds more like a clean echo, whereas the IIR filter sound more tin-canny. The sound of the IIR filter is repeated multiple times, and sounds more like reverberation than a single echo.

D. 3.1

Creating a tone complex shows us the difference that adding phase makes. Adding a random phase is adding a different "amount" of cosine and sine to each signal resulting in a seemingly more erratic graph. These 3 signals don't have a huge audible difference, especially at such a high pitch. When the pitch is lowered, a slight difference in tone can be heard. The sampling frequency `fs_s` limits the number of components can be added to the tone complex, at N=23 the highest frequency is 23kHz ($23 \cdot 1kHz$) which is higher that the Nyquist frequency ($\frac{fs\_s}{2} \approx 22kHz$).

E. 3.2

The synthesised piano sounded rather dull with only the fundamental frequency, and nothing like the tone on a piano. However the sound started to improve as we added some harmonic frequencies to the fundamental frequency. Adding phase, which meant that the Fourier transforms were now complex instead of purely real (for cosines) or purely imaginary (for sines), gave some depth to the sound but not as much as adding the harmonics. The sound is very similar to the original but is missing certain factors that would make it sound authentic, such as reverb and a gradually attenuating sound.

IV. Discussion

This report has showed the complexity of the different factors that are needed to be taken into account when analysing or recreating a signal or sound. All of these exercises could be combined to recreate a more authentic sounding synthesised piano: an IIR filter could be used to recreate the reverb sound that could be heard in the original sound. Furthermore, the impulse response of the piano could be found and could be convolved with different frequencies to create different notes. Potentially, a notch-filter could be used if one of the frequencies produced is the piano's resonant frequency. The pole-zero plot of the original sound could also be analysed to see if the piano acts like a low-pass filter, amplifying the lower frequency harmonics, producing a more bassy sound, or as a high-pass filter producing more treble. This could be used to optimise pianos, making them produce the exact amount of bass and treble desired.