31606 - Assignment number 1

101010, Frederik Ettrup Larsen (s163920), Pelle Michael Schwartz (s147170) and Irene Marie Malmkjaer Danvy (s163905)

I. Summary and objectives

This assignment asseses the uses of discrete signal processing using Matlabs Fourier transfers and plotting generating sinusoids and synthetic signals. The signals are processed as an audio file and then imported again. It is shown that the $rect(x)$ in time domain corresponds to a $sinc(\omega)$ in the frequency domain. Furthermore some matlab solutions are described and explained and lastly the Hands-on 2 excercise 1.3 is solved.

II. Methods

A. Two similar spectrums

In Matlab using a function to generate a sinusoid coupled with a function `make_spectrum` to make following plot:

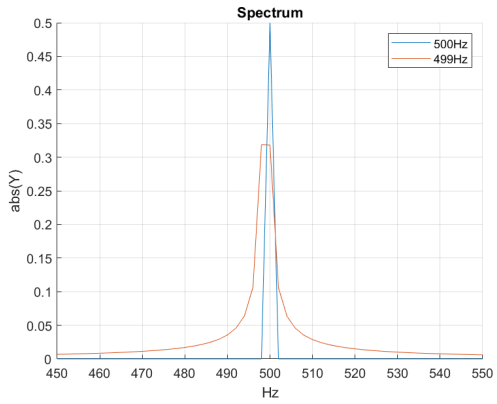

Figure 1. Matlab spectrum of a 500Hz and 499Hz sinusoid

The plot shows how, at the frequency of 500Hz, the `fft` returns a spectrum with clear max value at exactly, as it should. However, 499Hz shows a spectrum with a correct peak, but with slopes on each side. This does not reflect the actual spectrum of the signal. This shows that the `fft` does not always result in a complete rendition of the spectrum.

B. Fourier Transform of synthetic signal

The generated signal can be seen plotted in figure 2.

1) Plotting the generated signal: The plottet signal can be seen in figure 2

2) Various other plots: Some of the sinusoids are going to become sine functions, for instance for $k = 2$ were we get $\cos(8\pi \cdot f_0 \cdot t + \frac{2}{3}\pi) = \sin(8\pi \cdot f_0 \cdot t + \frac{1}{6}\pi)$, while others are going to become negative cosine functions, for instance for $k = 3$ we get $\cos(16\pi \cdot f_0 \cdot t + \pi) = -\cos(16\pi \cdot f_0 \cdot t)$.

All of the function can be converted to complex expressions, using Euler's identity, which states that

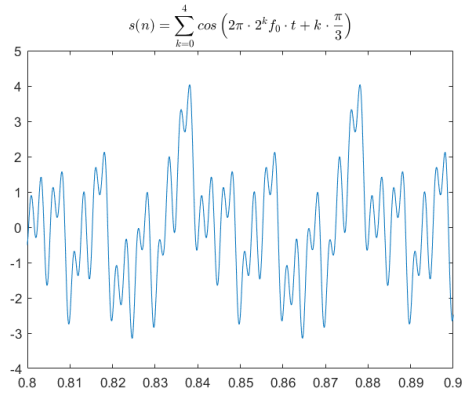

Figure 2. A plot of the generated signal plotted in [0.8s,0.9s]-

$$\cos(x) = \frac{e^{j \cdot x} + e^{-j \cdot x}}{2}$$

$$\sin(x) = \frac{e^{j \cdot x} - e^{-j \cdot x}}{2j}$$

With this conversion, the sum becomes:

$$s(n) = \sum_{k=0}^{4} \frac{e^{j \cdot (2\pi \cdot 2^k f_0 \cdot t + k\frac{\pi}{3})} + e^{-j \cdot (2\pi \cdot 2^k f_0 \cdot t + k\frac{\pi}{3})}}{2}$$

Which is by far a much uglier expression, it does however allow us to plot the real and imaginary parts of the function, as seen in figure 4 and to plot the phase and magnitude as seen in figure 3.

Here it should be noted that we can quite clearly see the various frequencies that the sinusoid is composed of, as every plot has some form for spike at the relevant frequencies. Most notably, we can se that the phase for 200Hz behaves differently from the other frequencies. This is due to that signal being a sine function instead of a cosine function like the others.

3) Plotting the first peak: The plot can be seen in figure 5, left side.

4) Storing and loading signals: When this signal was stored, matlab gave the warning `Warning: Data clipped when writing file.` This warning comes because we're trying to save the file 16 bits per sample. In this format, matlab cannot handle magnitudes above $|1|$, which our signal is, this leads to the signal being cut off. This can be avoided by dampening the signal down so the highest peak becomes 1, storing it. When loading it again, one would have to amplify it back to get the same signal as before. Our plot can be seen in figure 5, right side

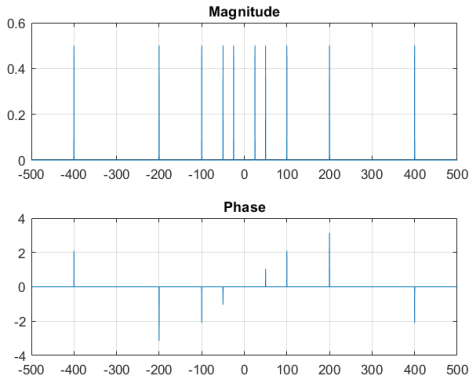

Figure 3. The magnitude and phase of the sinusoid signal

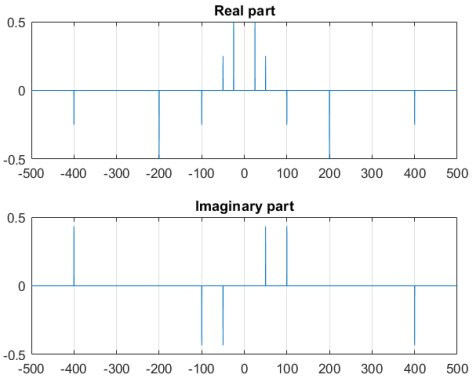

Figure 4. The real and imaginary part of the sinusoid

C. Some analytical work

In the following subsections the rect-function and the sinc-function are defined. The $rect$ function in the time domain is Fourier transformed in continous time and the $sinc$-function in the frequency domain is inverse Fourier transformed. Finally the $rect_W$ function is defined and Fourier transformed, and an expression for the resulting $sinc_W$ function's zero-crossings is defined.

1) Fourier transform of rect function: The $rect$ function is given as follows:

$$rect\left(\frac{x}{\tau}\right) = \begin{cases} 0 & |\, x \,| > \frac{\tau}{2} \\ 1 & |\, x \,| < \frac{\tau}{2} \\ \frac{1}{2} & |\, x \,| = \frac{\tau}{2} \end{cases}$$

We then Fourier transform it in continous time as follows:

$$\begin{aligned} F(\omega) &= \int_{-\infty}^{\infty} rect(\frac{t}{\tau}) \mathrm{e}^{-j\omega t} \mathrm{d}t = \int_{-\frac{\tau}{2}}^{\frac{\tau}{2}} \mathrm{e}^{-j\omega t} \mathrm{d}t \\ &= \left[\frac{1}{-j\omega} \mathrm{e}^{-j\omega t}\right]_{-\frac{\tau}{2}}^{\frac{\tau}{2}} = \frac{1}{-j\omega}\left(\mathrm{e}^{-j\omega\frac{\tau}{2}} - \mathrm{e}^{j\omega t\frac{\tau}{2}}\right) \\ &= \frac{2\sin(\frac{\omega\tau}{2})}{\omega} = \tau \frac{\sin(\frac{\omega\tau}{2})}{\frac{\omega\tau}{2}} = \tau \mathrm{sinc}\left(\frac{\omega\tau}{2}\right) \end{aligned}$$

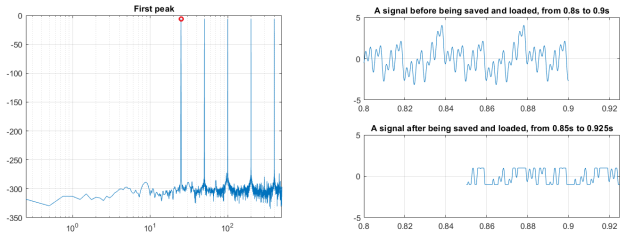

Figure 5. The first peak, marked with a circle, and the two signals before after being loaded, displayed at the different time intervals

It is clear that if the $rect$ function had been of height $\frac{1}{\tau}$ instead of 1, then the Fourier transform would have been $\mathrm{sinc}\left(\frac{\omega\tau}{2}\right)$.

2) Inverse Fourier transform of sinc function :

$$\begin{aligned} f(t) &= \frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{\sin(\frac{\omega\tau}{2})}{\frac{\omega\tau}{2}} \mathrm{e}^{j\omega t} \mathrm{d}\omega \\ &= \frac{1}{\pi}\int_{0}^{\infty} \frac{\sin(\frac{\omega\tau}{2})(\cos(\omega t) + j\sin(\omega t))}{\frac{\omega\tau}{2}} \mathrm{d}\omega \end{aligned}$$

Using the relations

$$\begin{aligned} \sin(a)\cdot\sin(b) &= \frac{1}{2}(\cos(a-b) - \cos(a+b)) \\ \sin(a)\cdot\cos(b) &= \frac{1}{2}(\sin(a-b) + \sin(a+b)) \end{aligned}$$

We get

$$\begin{aligned} &= \frac{1}{2\pi}\int_{0}^{\infty} (\sin\left(\omega\left(\frac{\tau}{2} - t\right)\right) + \sin\left(\omega\left(\frac{\tau}{2} + t\right)\right) \\ &\quad + j\left(\cos\left(\omega\left(\frac{\tau}{2} - t\right)\right) - \cos\left(\omega\left(\frac{\tau}{2} + t\right)\right)\right)\frac{2}{\omega}\mathrm{d}\omega \end{aligned}$$

$$\begin{aligned} &= \frac{1}{2\pi}\int_{0}^{\infty} \frac{2j\mathrm{e}^{-j\omega(\frac{\tau}{2}-t)} - j\mathrm{e}^{-j\omega(\frac{\tau}{2}+t)}}{\omega}\mathrm{d}\omega \\ &= \frac{1}{2\pi}\int_{0}^{\infty} \frac{2}{j\omega}\left(\mathrm{e}^{-j\omega(\frac{\tau}{2}+t)} - \mathrm{e}^{-j\omega(\frac{\tau}{2}-t)}\right)\mathrm{d}\omega \end{aligned}$$

Remembering that

$$sgn(x) = \begin{cases} -1 & x < 0 \\ 0 & x = 0 \\ 1 & x > 0 \end{cases} = \frac{1}{2\pi}\int_{-\infty}^{\infty} \frac{2}{j\omega}\mathrm{e}^{j\omega t}\mathrm{d}\omega$$

We get

$$= sgn\left(t + \frac{\tau}{2}\right) - sgn\left(t - \frac{\tau}{2}\right)$$

Looking at the definition of sgn, we can show, that this is indeed the $rect$ function:

$$sgn(t+1) - sgn(t-1) = \begin{cases} 0 & t < -\frac{\tau}{2} \\ 1 & -\frac{\tau}{2} < t < \frac{\tau}{2} \\ 0 & 1 < \frac{\tau}{2} \\ \frac{1}{2} & t = -\frac{\tau}{2} \\ \frac{1}{2} & t = \frac{\tau}{2} \end{cases}$$

or

$$sgn(t+1) - sgn(t-1) = \begin{cases} 0 & |t| > \frac{\tau}{2} \\ 1 & |t| < \frac{\tau}{2} \\ \frac{1}{2} & |t| = \frac{\tau}{2} \end{cases}$$

Which is the $rect$ function, as previously defined.

3) An expression of the zero-crossings of the a sinc function corresponding to the $rect_W$ function: We have the $rect_W(x)$ function, defined as:

$$rect_W(x) = \begin{cases} \frac{1}{W} & 0 \leq x < W \\ 0 & x \geq W \end{cases}$$

We can find the corresponding $sinc$ by using what we have just learned about Fourier transform and what we already know about the phase shift property

Since $rect_W(t)$ is time-shifted $W/2$ along the time axis compaired to the $rect(\frac{t}{W})$, the fourier transformation has to be multiplied by $\mathrm{e}^{-j\omega\frac{W}{2}}$ in accordance with the time shift property of Fourier transformation.

Also worth noting is that $rect_W(t)$ gives a rectangle of height $\frac{1}{W}$, meaning instead of the expected $F(rect(\frac{t}{W})) = W\mathrm{sinc}(\frac{\omega W}{2})$ we will simply get $F(rect_W(t)) = \mathrm{e}^{\frac{-j\omega W}{2}}\mathrm{sinc}(\frac{\omega W}{2})$.

Since $\sin(x) = 0$ whenever $x = n\pi, n \in \mathbb{Z}$, then it must also hold that $\mathrm{sinc(x)} = \frac{sin(x)}{x} = 0$ when $x = n\pi, n \in \mathbb{Z}$. Therefore the zero crossings of the $sinc$-function are at $\frac{\omega W}{2} = n\pi, n \in \mathbb{Z}$ giving them a length of $\frac{\omega W}{2\pi}$.

D. How and why

1) make_spectrum: The `make_spectrum` function is made to return the complex spectrum of a signal in the shape of two one-dimensional vectors containing coordinate pairs of frequency and magnitude. It is returned with both the positive and negative spectrum. The code for it looks as follows:

```
function [Y, freq] =
            make_spectrum(signal, fs)
Y = fftshift(fft(signal));
Y = Y/length(Y);
delta_f = fs/length(signal);
freq  = -fs/2:delta_f:fs/2-delta_f;
end
```

the function `fft` results in an array which is limited by positieve/negative nyquist frequency. This however is why `fftshift` is called to shift the latter part as the first elements, effectively swapping out the fist and second half of the array. `delta_f`, which is the increment of frequency for each value, is inherently from the `fft` the sampling frequency divided by the number of samples. Now the `freq` spectrum can be made which goes from the negative/positive frequency with steps of `delta_f`.

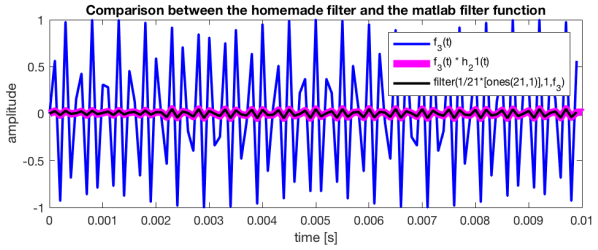

Figure 7. A comparison between the output of filtering $f_3(t)$ with an order 21 moving average filter by convolving it with the impulse response, and using matlab's built in filter function. It is clear the output is the same, probably because the math being done is the same despite the different commands typed in.

E. Hands-On 2, 1.3

An order 21 moving average filter has 21 delayed inputs, which means the impulse response is simply 21 sequential δ pulses, each attenuated down to a height of 1/21th. If the impulse response has a frequency of 10kHz and a time vector from $[0:100]$ ms, it is simply a vector of length 1000 whose first 21st elements are 1's and the rest are zeros.

$$f_1(t) = \sin(2\pi(500\mathrm{Hz})t) \quad (1)$$
$$f_2(t) = \sin(2\pi(2200\mathrm{Hz})t) \quad (2)$$
$$f_3(t) = \sin(2\pi(4050\mathrm{Hz})t) \quad (3)$$

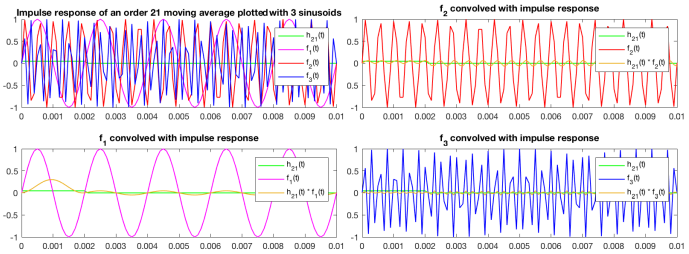

Figure 6. The impulse response is plotted with the three sinusoids in equation 1,2 and 3 above. Also shown is each signal convoluted with the impulse response.

It is clear that convolving the signals with the running average filter is the same as filtering them with the matlab filter function, as can be seen in figure 7.

On figure 6 it is clear that the filtered signals are attenuated, and that during the first 2 ms they are differently attenuated. This is because the moving average filter relies on feedback; it gives the average of the current value and the 20 that came before it. Before the 21st value, this is not possible, as there aren't yet 20 values in memory. Therefore, for the first 2 ms, the filter works as a cumulative filter that divides each successive sum with 21.

The higher frequency signals are more attenuated by filtering. This is because our signal isn't displaced along the

amplitude (y) axis, which means the average of a full period, or several full periods, is zero. The more periods are averaged (or the larger the part of a period, if the period is longer than the time we're averaging over) the smaller said average will be, which is why all moving average filters are low pass filters. In keeping with above described phenomenon, the signal most attenuated by filtering is the signal with the highest frequency: $f_3(t)$.

F. Analytical calculation of convlution

A sinusoid with a frequency of 10Hz, with a sampling frequency of 50Hz, could look like this, if sampled for one tenth of a second:

$$\begin{aligned} f(u) = \{&0 \quad 0.95 \quad 0.59 \quad -0.59 \quad -0.95 \\ &0 \quad 0.95 \quad 0.59 \quad -0.59 \quad -0.95\} \end{aligned}$$

A 5th order running sum filter would look like this

$$h(u) = \{1 \quad 1 \quad 1 \quad 1 \quad 1 \quad 0 \quad 0 \quad 0 \quad 0 \quad 0\}$$

If we say that u goes from 1 to 10 in the convolution we get:

$$\begin{aligned} y(k) &= \sum_{u=1}^{k} f(u) \cdot h(k-u) \\ y(0) &= f(0) \cdot h(0) = 0 \\ y(1) &= f(0) \cdot h(1) + f(1) \cdot h(2) = 0.95 \\ y(2) &= f(0) \cdot h(2) + f(1) \cdot h(2) + f(2) \cdot h(0) = 1.54 \\ &\vdots \\ y(14) &= f(9) \cdot h(5) = -0.95 \end{aligned}$$

The validity of this convolution was tested through this matlab code:

```
[t,s] = generate_sinusoid(1,10,0,50,0.2);
conv(ones(5,1),s(:))
```

The $h(u)$ function is essentially a rect-function, which is, as we know, an ideal lowpass filter. For this reason, the signal is completely dampened, except for in the start and the end where the system charges and discharges so to say. the two signals can be seen in fig. 8

Let p be the order of a running average or sum filter with sampling frequency f_s, filtering out signals not transposed along the amplitude-axis. Then all frequencies blocked out will be $f = \frac{nf_s}{p}, n \in \mathbb{N}, n \neq 0$. Thus the running average, with a sampling frequency of 10kHz, blocks out the following frequencies:

$$\begin{aligned} f_{21,blocked} &= \frac{n10 \cdot 10^3}{21}, n = 1, 2, 3... \\ &= \frac{10000n}{21}, n = 1, 2, 3... \end{aligned}$$

Meaning the moving average filter blocks out 476.1905 Hz, 952.3810 Hz 1428.6 Hz and so on.

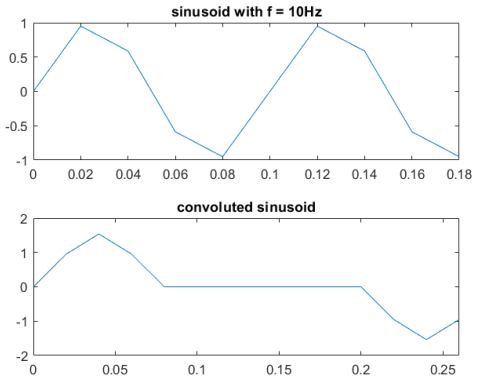

Figure 8. A sinusoid and the convolution of sinusoid with a running sum filter.

Likewise the moving sum filter of order 5 with a sampling frequencies of 50 Hz blocks out the following frequencies:

$$\begin{aligned} f_{5,blocked} &= \frac{n50}{5}, n = 1, 2, 3... \\ &= 10n, n = 1, 2, 3... \end{aligned}$$

Meaning it blocks out 10Hz, 20Hz, 30Hz and so on.

The reason this formula works, is that if a multiple of exactly full periods are held within the samples the sum/average is taken over, then they will cancel each other out, the result will be zero and the frequency filtered out.

III. Results

We now know that a matlab does not always display a complete reflection of the discrete signal as a spectrum. It is also proven how a *sgn* is transformed to a *rect* function from time to frequency domain and vice versa. Different matlab functions have been used to produce informative plots. Comparisons between a homemade filter and matlab's built-in `filter` function show that they are effectively the same. Lastly it is shown how a convolution with a single sine wave behaves in discrete time.

IV. Discussion

Problems were solved mainly using matlab functions. The fft function in matlab creates some issues, but works as intended, however requiring some extra work to utilize it properly, this must be done carefully to avoid errors. Accurate and informative plotting of functions and spectrums in this exercise is important to communicate and explain the given function/spectrum.

V. How much time did we use

We have each put about 20 hours of work into this report