Department of Physics
Umeå Universitet March 29, 2015

Numerical Methods in Physics
Fast Fourier Transforms

Jesper Vesterberg (jeve0010@student.umu.se)

March 29, 2015

Numerical Methods in Physics
supervisor: Peter Olsson

Jesper Vesterberg (jeve0010@student.umu.se)

1 The Fourier Transform of a Gaussian

For this assignment we look at at the fast fourier transform of the Gaussian

$$x(t) = \frac{1}{\sqrt{\pi\alpha^2}} e^{-t^2/\alpha^2} \tag{1}$$

and compare it with its true theoretical Fourier transform

$$X(f) = e^{-\frac{1}{4}(2\pi\alpha f)^2}. \tag{2}$$

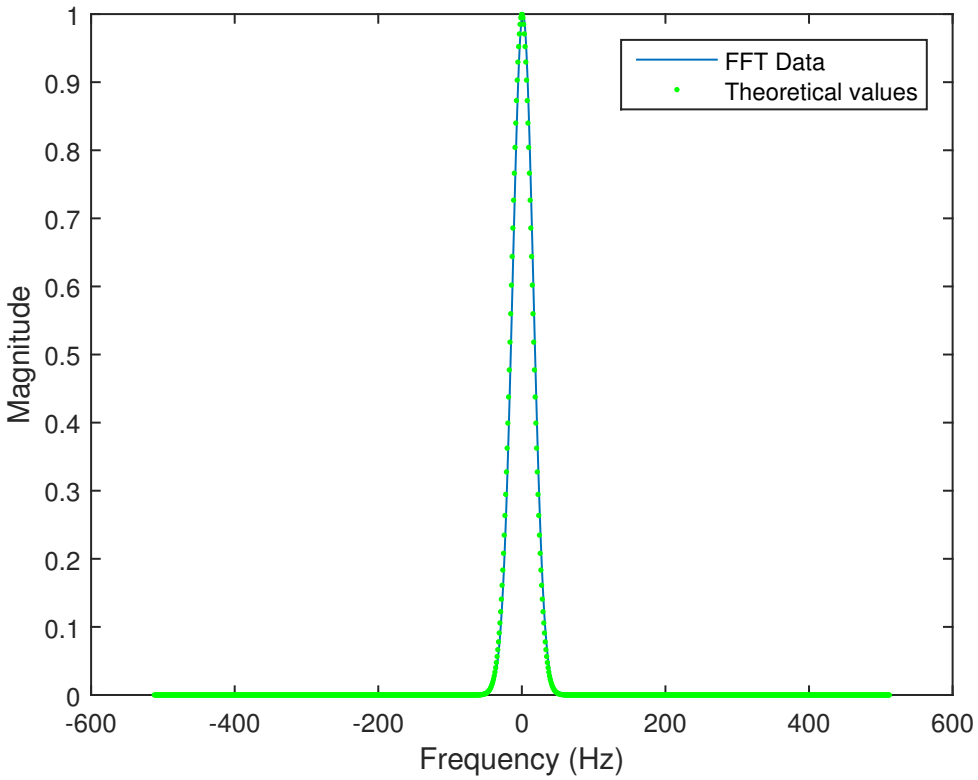

Figure 1 – Comparison between the normalized fast fourier transform and the true theoretical fourier transform of the Gaussian (eq. 1)

As we can see in figure 1 the fastc fourier transform seem to be pretty spot on. The resolution of the frequencies dependent on the sampling frequency and the amount of samples taken. This because the amount of samples taken directly translates on to how many different frequency values we can get out of the transform[1]. The sample frequency tells us which frequencies we could possibly have caught since one needs two points to capture a period. Basically we have frequencies in the range $[-\frac{1}{2\Delta}, \frac{1}{2\Delta}]$, where Δ is the

[1]Due to the linearity of the transform we get one output for each input

time between samples and since we simulate that we take 1024 samples we get

$$(\frac{1}{2\Delta} + \frac{1}{2\Delta})/\textit{number of jumps between frequency values} \tag{3}$$
$$=(\frac{1}{2\Delta} + \frac{1}{2\Delta})/1023 \tag{4}$$
$$=1024/1023 = 1.001. \tag{5}$$

This basically means we have 1.001 Hz between the frequencies we get checked in the fourier transform.

1.1 Running the code

The code relating to this exercise are all located in the directory

`~jeve0010/Documents/jesper/fnm/lab1/ex1`

To run the program doing the fft simple call `make` then execute `./fft` then to see the data plotted simply run the matlab script `plotty.m`. The relevant data points lies in imTrans and realTrans.

2 The Fourier Transform of a Gaussian

In this exercise we look at the equation

$$u(t) = \bar{u}sin(2\pi f_c t) \tag{6}$$

where $f_c = 8$ and $\bar{u}$ basically is a square wave going between the value 1 and a value u_1 which we vary through the exercise. The square wave changes values after a period of the sine function has passed. This is to simulate a message 01010101. Thus we plot the function for one whole time unit.

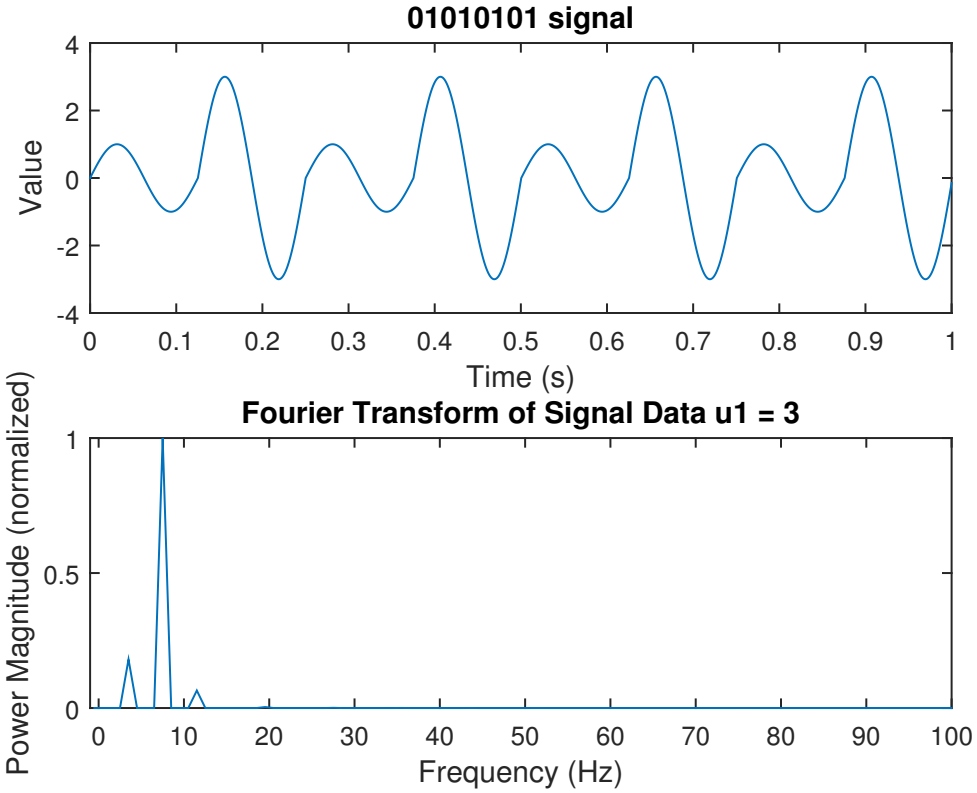

Figure 2 – Looking at the frequency spectrum of the signal with $u_1 = 3$.

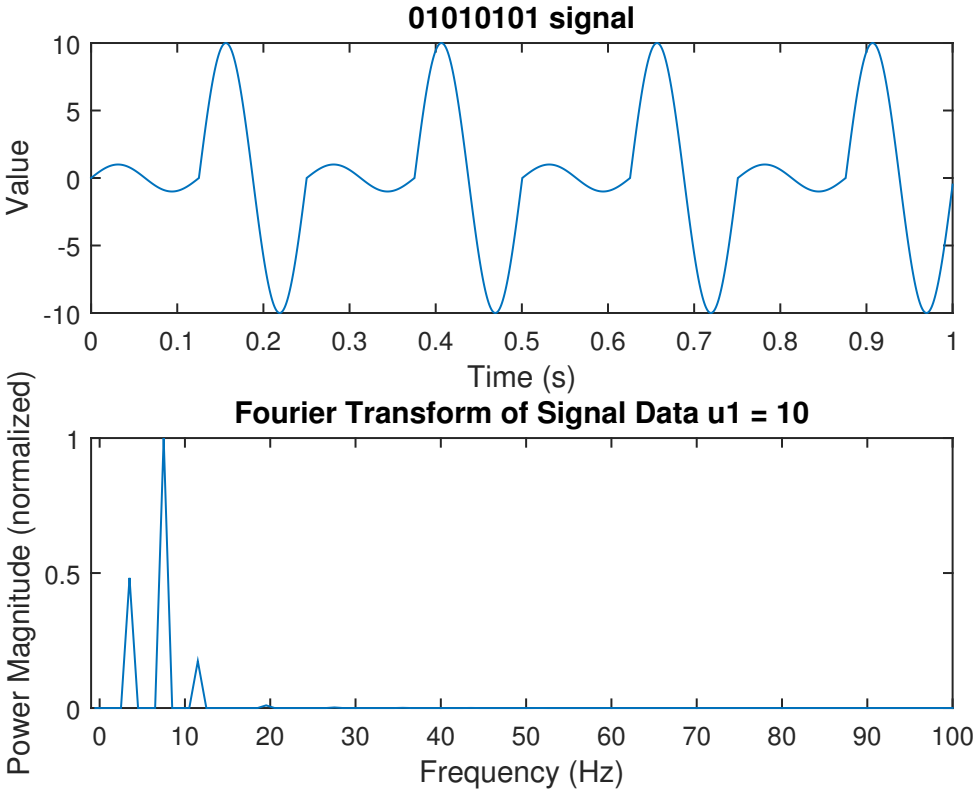

Figure 3 – Looking at the frequency spectrum of the signal with $u_1 = 10$.

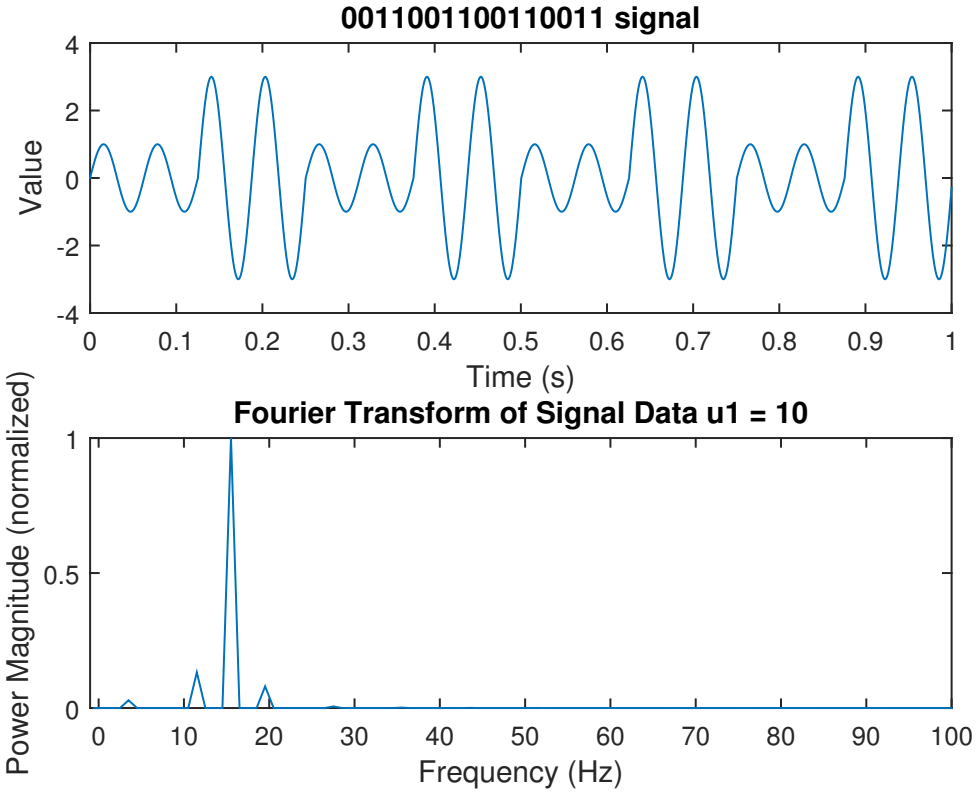

Figure 4 – Looking at the frequency spectrum of the signal with $u_1 = 3$. Ontop of that we doubled the carrier frequency without changing the modulation of the sine function. We can see that we have the same bandwidth but the same amount of information is given.

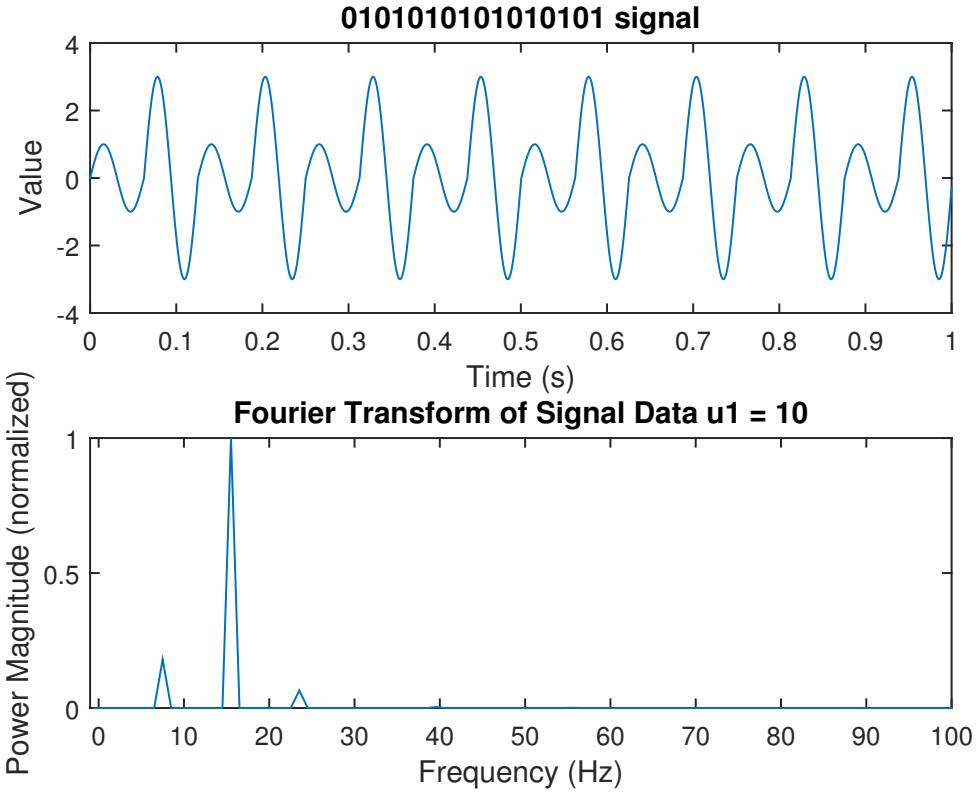

Figure 5 – Looking at the frequency spectrum of the signal with $u_1 = 3$. Ontop of that we double carrier wave frequency and the modulation of the sine function. We can see that we get double the information, at the cost of having a twice as large bandwidth.

Looking at figure 4 we can see that the frequency bandwidth is 8 Hz (if the time unit is in seconds). Looking at the discussion in the lab instruction about equation (7) in that paper we could perhaps guess the behavior of having the carrier frequency in the middle of the bandwidth. This because of the centered carrier wave frequency in that case. Here f_b gets to be the square wave instead. That said... it feels slightly like its not a complete connection... only a vague similarity.

If we increase the value of u_1 from 3 to 10 we can see in figure 5 that the peaks that corresponds to the square wave gets larger in comparison to the peak that corresponds to the carrier wave. This seems natural since there is a larger effect of the square wave, thus it should get a larger part in the corresponding fourier transform.

Its interesting to see that if we double the carrier wave frequency and want to send zeros followed by ones, but still be constricted to the same bandwidth, we cant do it. This because if we modulate the sine function faster we get a larger bandwidth. We could however send 0011001100110011 instead of 01010101, but this is not of much interest since its basically the same information. Basically if we change the modulation of the sine function the bandwidth changes. If we increase the modulation we get a larger bandwidth, but can send more information in the same time. If we decrease the modulation we get a smaller bandwidth, but can send less information in the same time. This also seem to depends on each other linearly.

On a side note one could say that you get double the information since you could

count a double zero as a number 2 and a double 1 as a number 3. Ontop of that you could go to 3,4,5,6 zeros and ones and let it signify different values, while still be within the bandwidth. Thus you could push more information through the same bandwidth requirement, by only changing the carrier frequency.

2.1 Running the code

The code relating to this exercise are all located in the directory

`~jeve0010/Documents/jesper/fnm/lab1/ex2`

To run the program doing the fft simple call `make` then execute `./amfft` then to see the data plotted simply run the matlab script `plotty.m`. The relevant data points lies in imTrans and realTrans.

3 Extracting Information from a Noisy Signal

In this exercise we simply wanna apply the given gaussian filter onto a given noisy signal. Using this we then wanna decode a bit message in the noisy signal. The signal was given in the file `am26.dat`, and the message was decoded to be `FNM15|jg;.""HAD`. In figure 6 we can see how the signal looks in the fourier space before and after the filter was applied. In figure 7 we can see how large an impact the filter has onto the original signal.

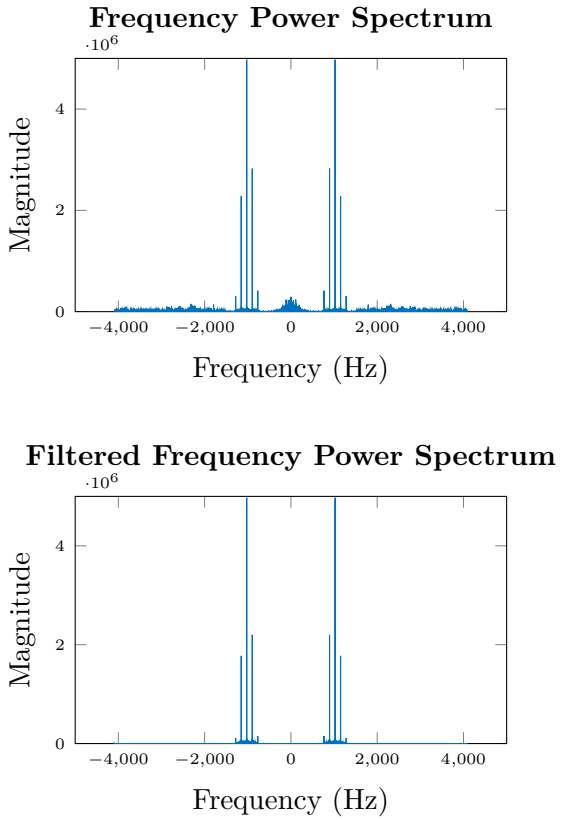

Figure 6 – Here we clearly depict the difference in the fourier space before and after the filtering.

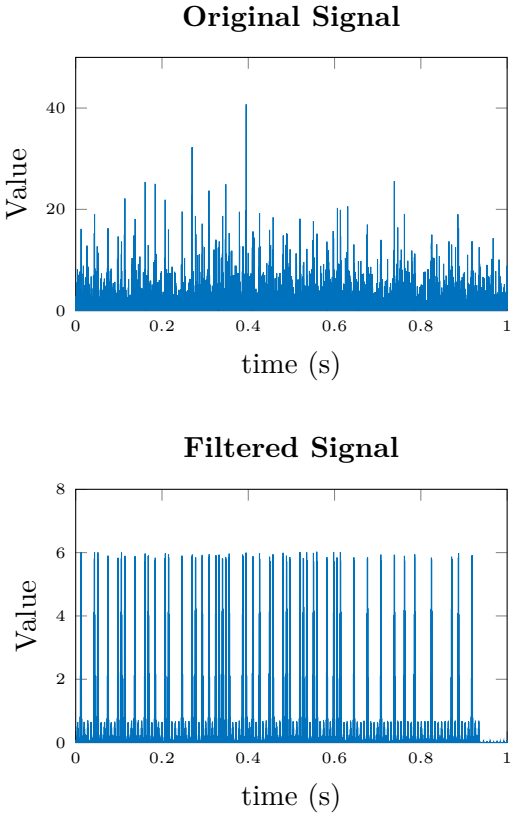

Figure 7 – Here we clearly depict the difference in the time space before and after the filtering.

3.1 Running the code

The code relating to this exercise are all located in the directory

`~jeve0010/Documents/jesper/fnm/lab1/ex3`

To run the program doing the fft simple call `make` then execute `./decode [message to decode]` then to see the data plotted simply run the matlab script `plotty.m`.