Digital Signal Processing SS 2022: 5. Assignment

DFT, FFT and STFT

Group number 34

---

1. Task

a) Give frequency resolution

1 ms = 0.001 s in which 100 samples. Therefore, we have $f_s = \frac{100 \text{ samples} \cdot 100}{s}$ (which is number of samples per second).
DSP08, p. 56: $\Delta f = \frac{f_s}{N_{FFT}} = \frac{\frac{100000}{s}}{100} = \frac{1000}{s}$ or $\Delta f = \frac{1}{MeasurementTimeSec} = \frac{1}{0.001s} = \frac{1000}{s}$

b) period of DFT spectrum in terms of ...

... samples

We have a 100-point DFT/FFT. So, after 100 points in frequnecy domain we get a repetition of the pattern before. Therefore: T = 100 (samples)

... frequency

$f_0 = \frac{1}{T} = \frac{1}{100}$

... normalized angular frequency

DSP03, p. 8: $\Omega = \frac{2\pi f_0}{f_s} = \frac{2\pi \frac{1}{100}}{100000}$

c) Why 128 samples chosen?

We need 2^c where c is in this case 7. The reason is that we need a even number of inputs for the FFT because they are pairwaise combined to get the input for the next stage of the FFT procedure. Then it goes on in the same kind of pattern.
Furthermore, 128 is the closest upper option to 100 when we make use of 2^c.

d) Give new frequency resolution

When we make a 128-point DFT/FFT, then we get $\Delta f = \frac{f_s}{N_{FFT}} = \frac{\frac{100000}{s}}{128} = \frac{781.25}{s}$

e) Interpretation of changed frequency resolution

The points are now closer together in the DFT spectrum and there are more of them within the same interval. However, we added no further information because the added points are DFT curve restricted. We only get a nicer visualization when we make one.

2. Task

a) Select four signals from the given table. Convert them into frequency domain by using DFT. For this purpose make use of the file DFT_example.m

Chosen signals:

1. DC
2. Triangular pulse
3. cosine
4. cos^2-pulse

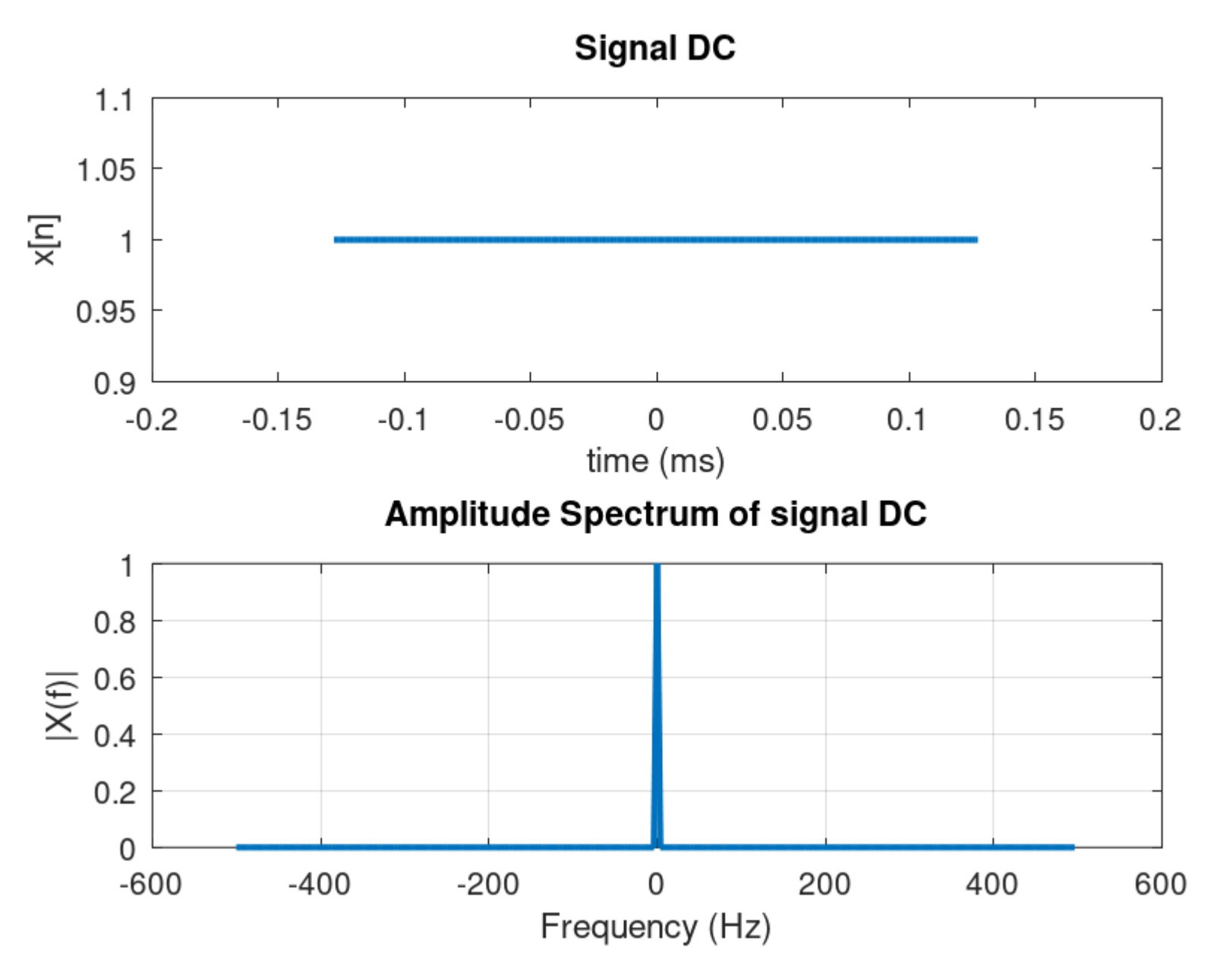
We will also show parts of our code here because they are integral parts of this task.

1. DC

Let's get started with a simple signal which is always one. Its spectrum should be a dirac function. For this purpose we used the code

```
DC = ones(N,1);
```

which creates a signal array of length N with only 1-s as elements. When converting the x-axis indices from sample indices into time we don't need to change here the values of DC because it stays always the same (is independent of time). The graphic gives us the correct spectrum for DC, the dirac function:



2. Triangular pulse

This pulse was more complicated to implement because we had to consider the x-axis convertion from sample indices n to time. Firstly, we created an array only containing 0-s:

```
triangular = zeros(N,1);
```

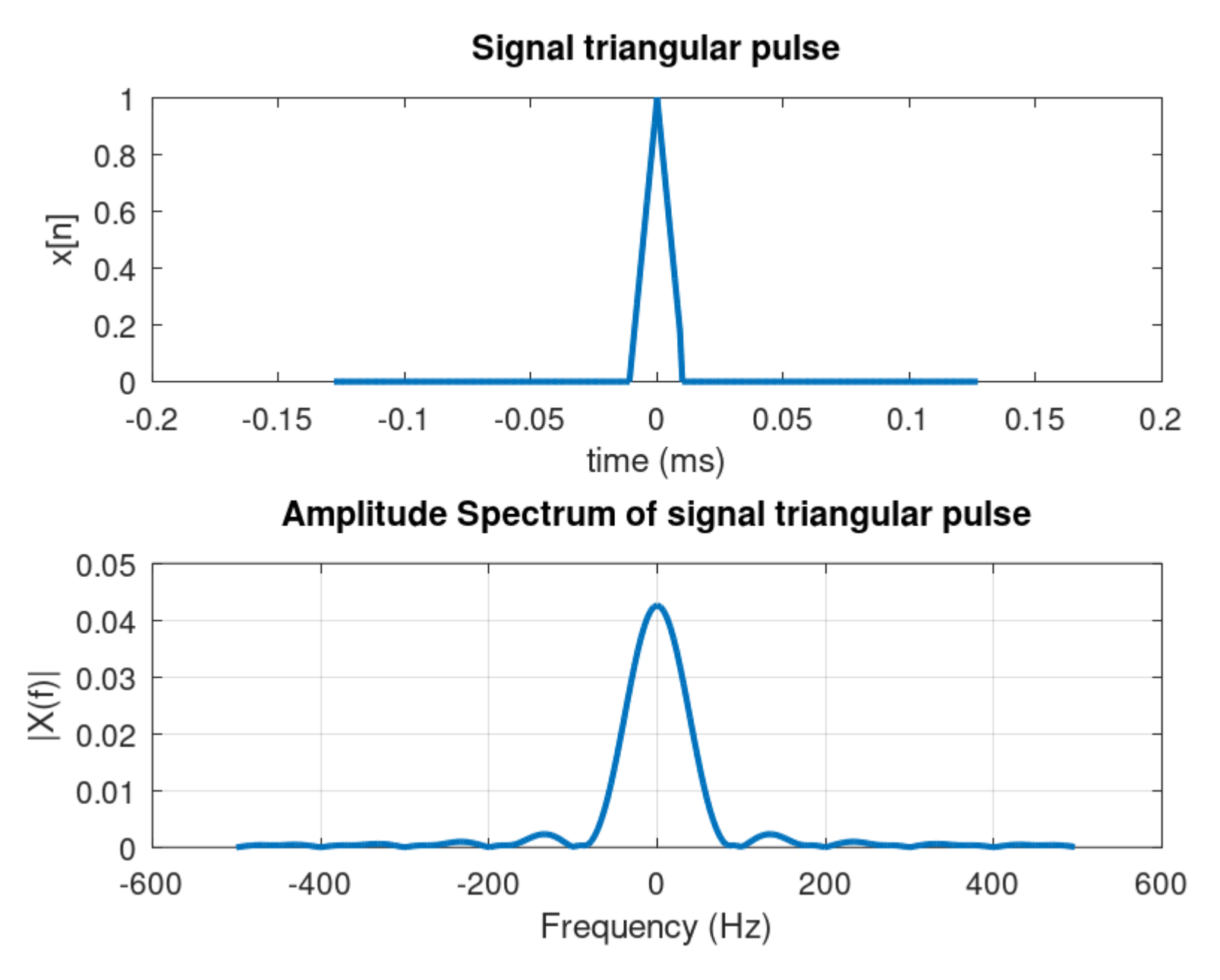
Then we looked at a window around half of the array. Here the pulse should get values forming the shape of a triangle. For this purpose, we looped over that area. The border of the area was described by **(not one to one code here)**

```
n_final_ind = n(N/2-10)/Fs;
```

where already the time transformation was done with "/Fs". During the looping process as mentioned above, we assigned new values where $triangular(currentIndex) = 1 - \frac{\text{absolute sample index time transfered}}{\text{border time transfered}}$, in our code:

```
triangular(index) = 1 - (abs(n(val)/Fs)/abs(n_final_ind));
```

(Attention: Only parts of code mentioned here. For the full picture simply have a look at DFT_example.m)

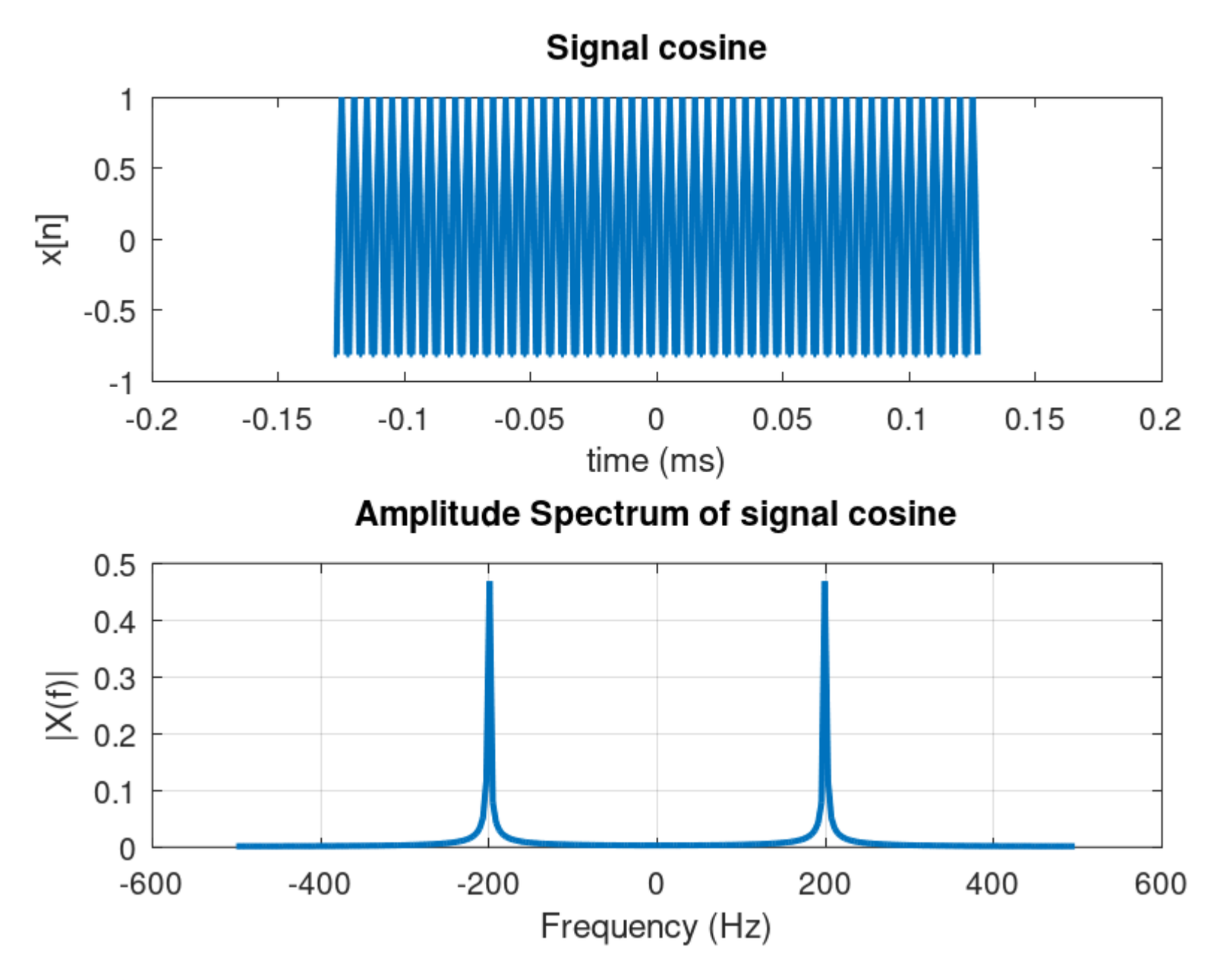


3. Cosine

Also here we considered the time transformation during creating the function:

```
cosine = cos(2*pi*f_0*(n/Fs));
```

where (n/Fs) = t (time) and f_0=200.
Below you can see that the peaks of our speactrum are exactly at +/- 200 as it should be because of f_0=200:

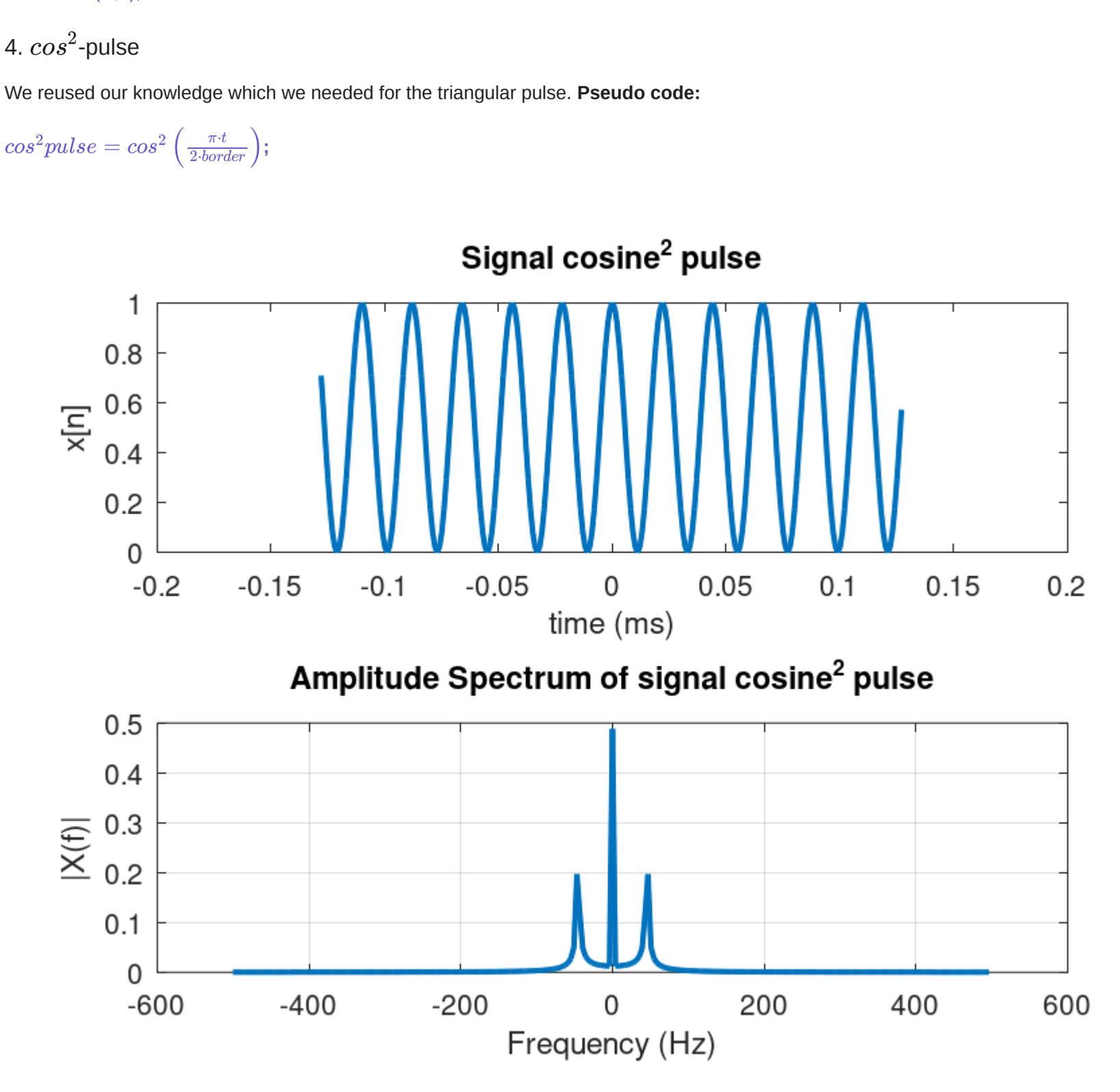



```
DC = ones(N,1);
```

4. cos^2-pulse

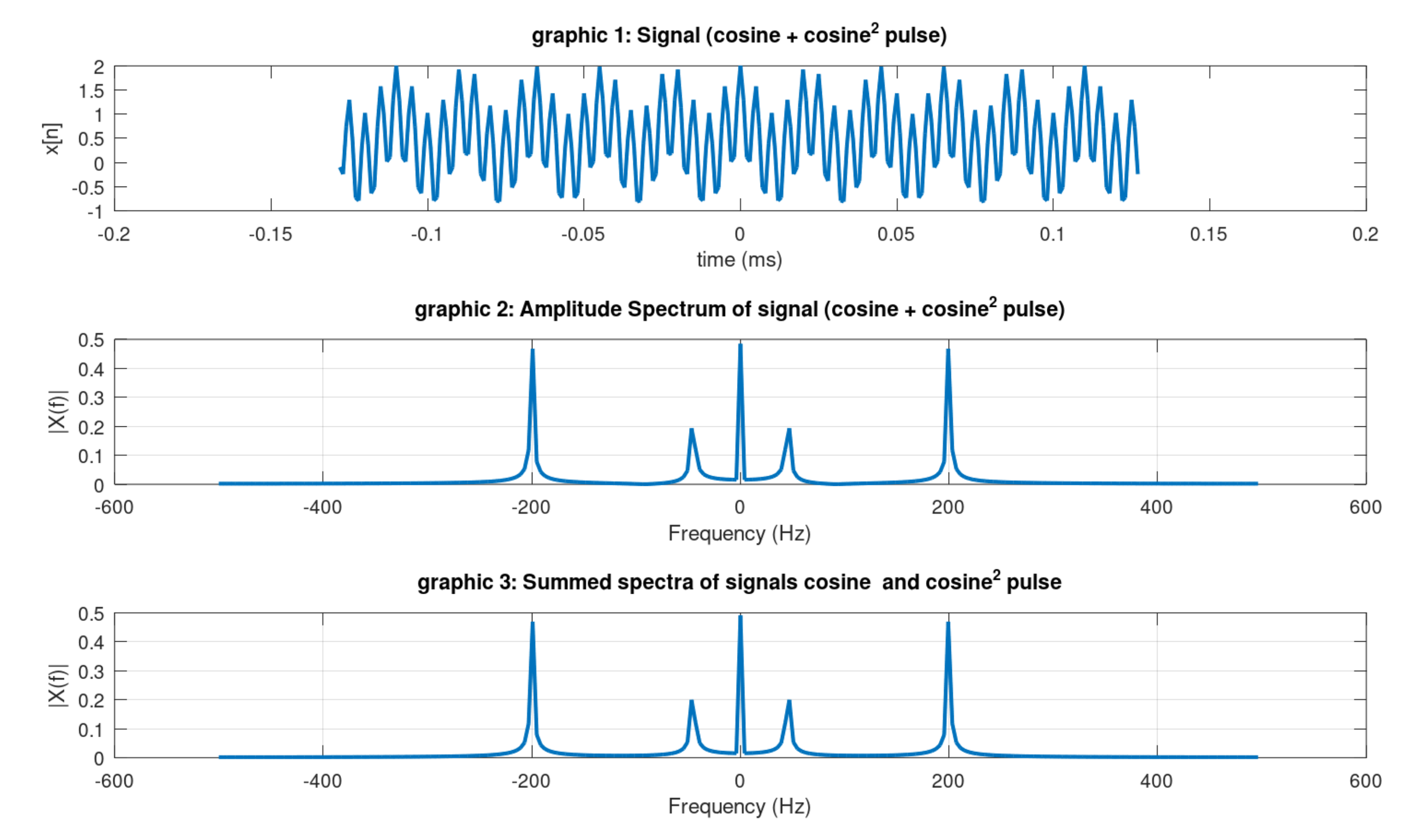
We reused our knowledge which we needed for the triangular pulse. **Pseudo code:**

$cos^2 pulse = cos^2\left(\frac{\pi \cdot t}{2 \cdot border}\right)$;



b) Showing linearity property of DFT by comparing the spectrum of two added signals with the sum of two spectra of two single signals where the same functions are used.

Remark for DFT_example.m: When you only want to make 2a) set the varaible task_b = true; to false. However, you also have to change the used signals when you want the ones from above.

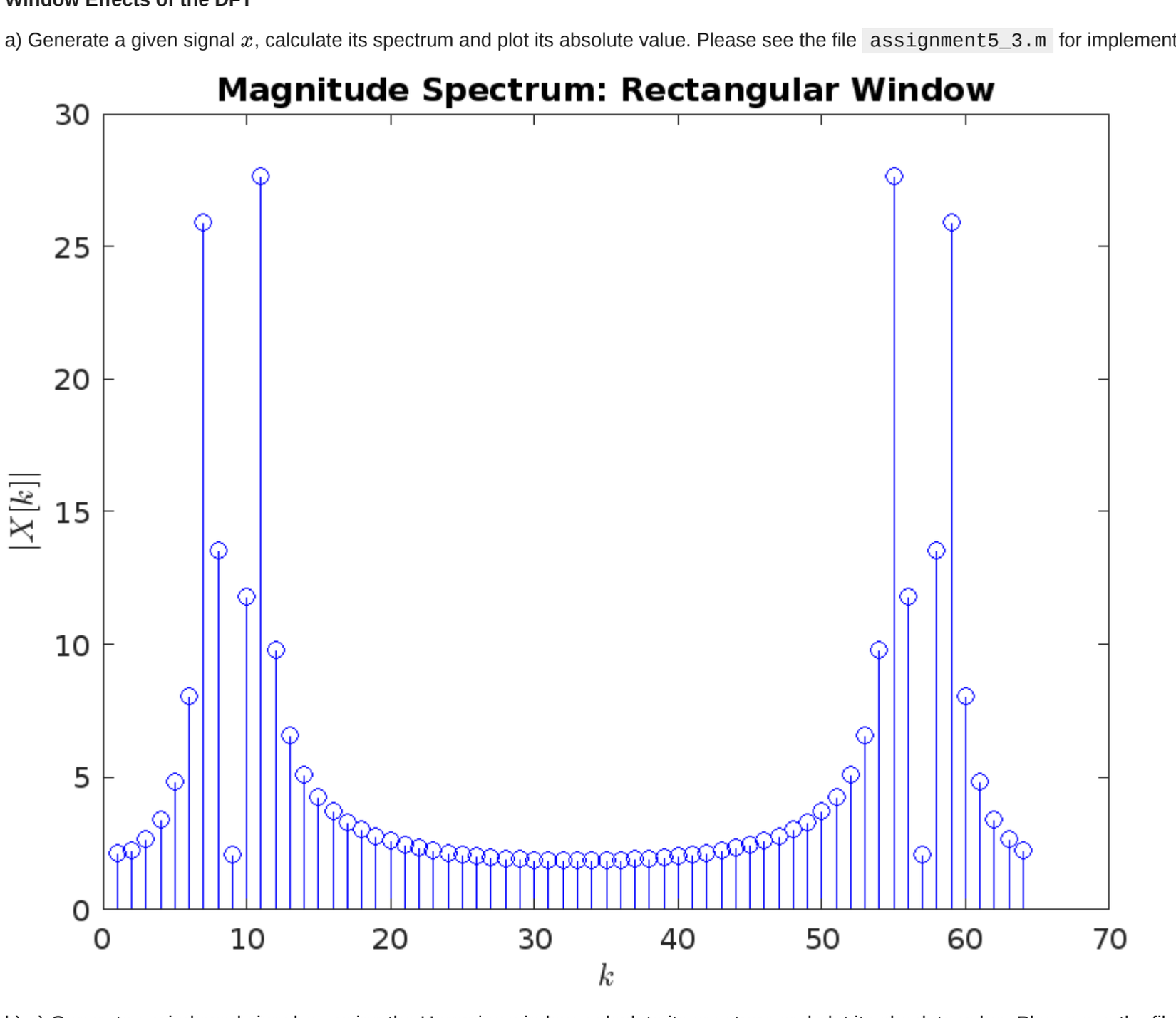


We created a png-file which shows us if the linearity porperty holds. We have chosen the signals cosine and $cosine^2 pulse$ because their spectra are easy to add up and the addition can be visually really easy seen by human eye.
graphic 1 simply shows us the new signal which is a sum of the two chosen signals in time domain.
graphic 2 gives us the spectrum of graphic 1.
graphic 3 is interesting now. For this graphic we converted the two signals separately into spectra. Then we summed up the two spectra. The graphic looks on first glance exaclty like graphic 2 which would support the linarity assumption of DFT.
To go sure we have graphic 4 which illustrates abs(graphic2-graphic3). So, here you can see how much the two spectra differ. The values of y-axis are very small. You can interpret them as rounding errors from resuting from the computations of graphic 2 and 3. Therfore, we can assume that linearity of DFT holds.

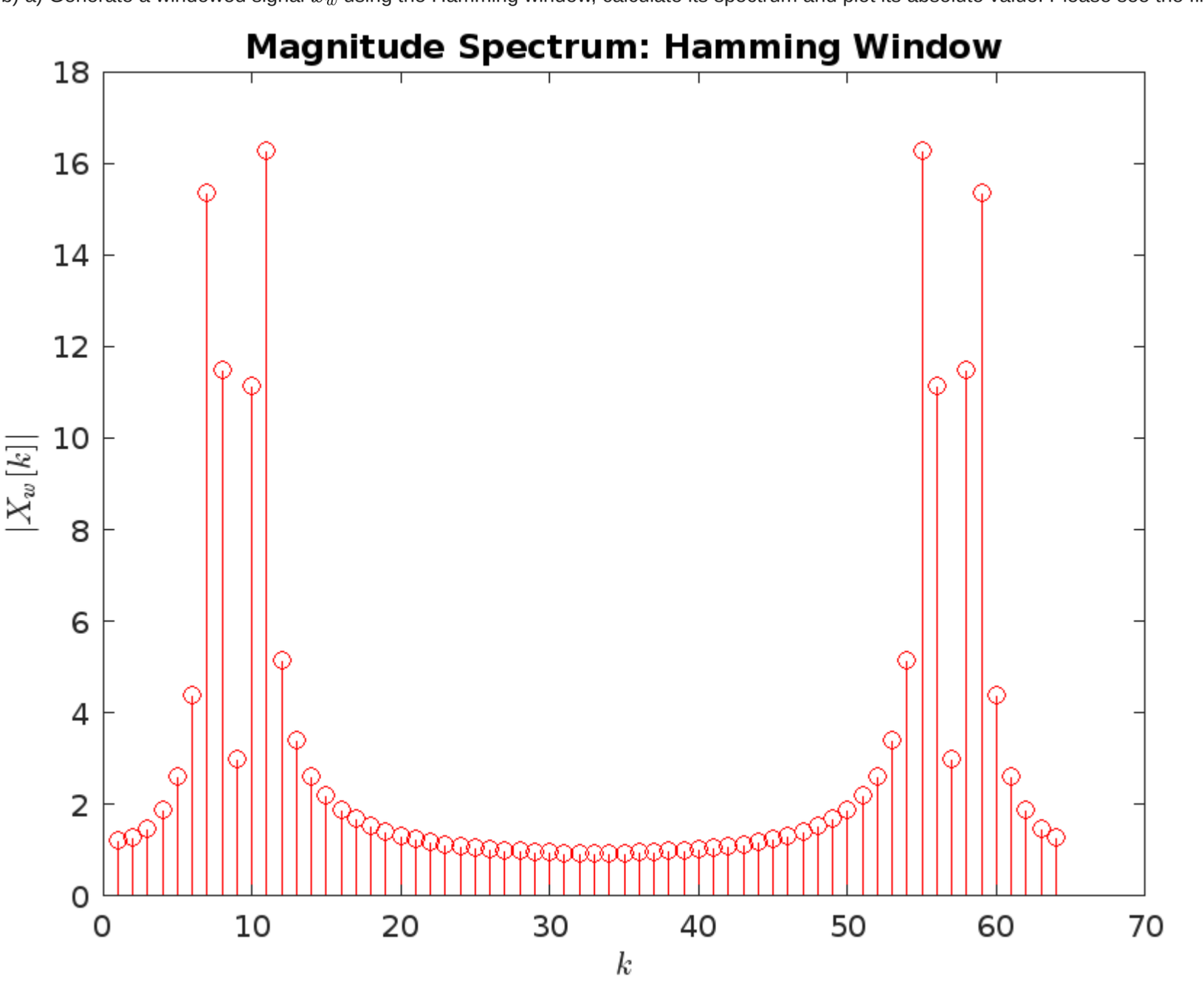
3. Task

Window Effects of the DFT

a) Generate a given signal x, calculate its spectrum and plot its absolute value. Please see the file `assignment5_3.m` for implementation details. The result is shown below:

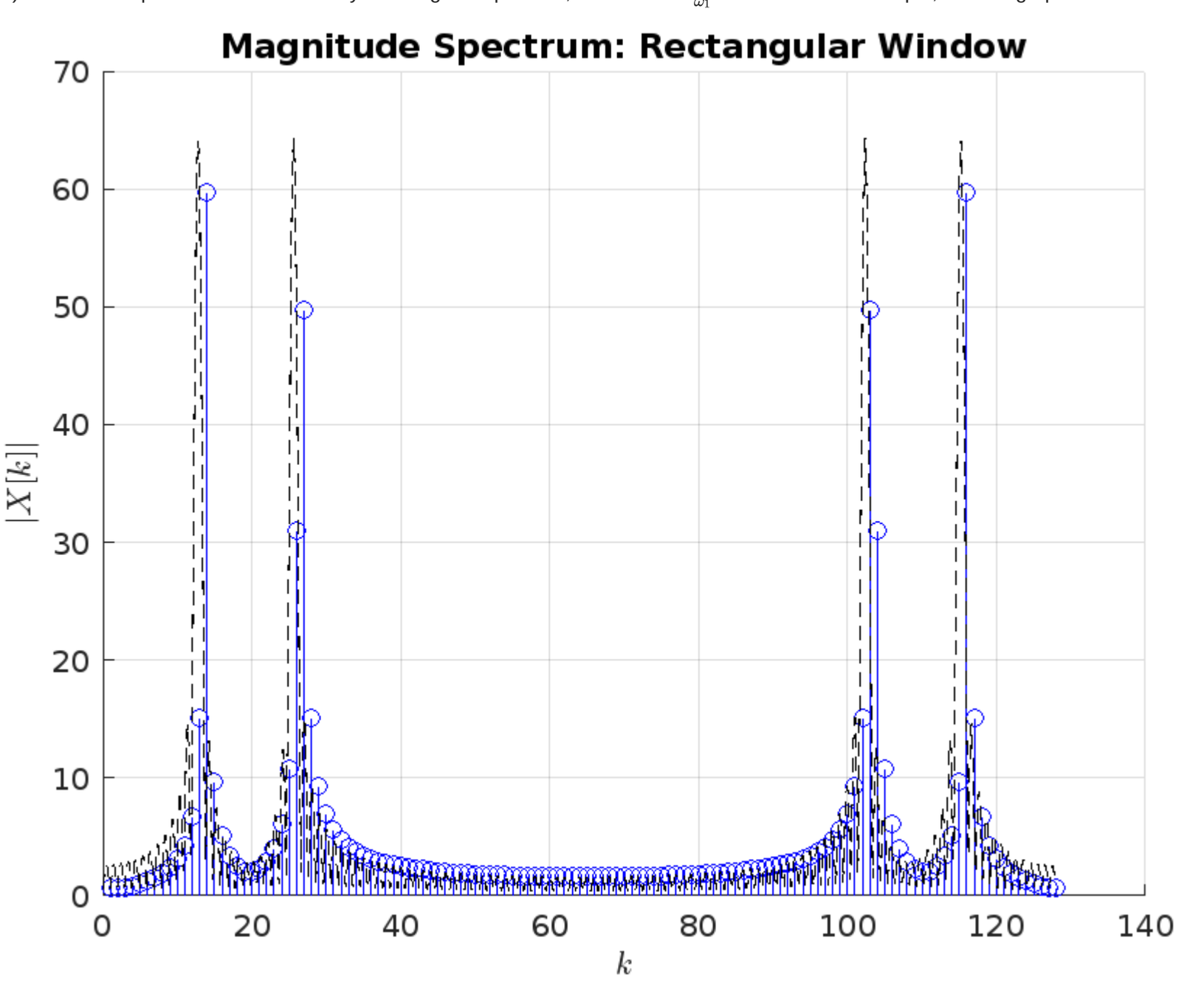


b) a) Generate a windowed signal x_w using the Hamming window, calculate its spectrum and plot its absolute value. Please see the file `assignment5_3.m` for implementation details. The result is shown below:



c) The Hamming window places the zero-crossing at frequency $\frac{5pi}{N-1}$ cancelling the first sidelobe of the Hann window, but resulting in a smaller height, as can be seen in the diagram. Compared to the rectangular windows, some frequencies are strechted.

d) The DTFT equals the DFT if and only if the signal is periodic, i.e. the ratio $\frac{\omega_2}{\omega_1}$ is rational. For example, see the graph below. In this case, the signal period and signal length are coprime.

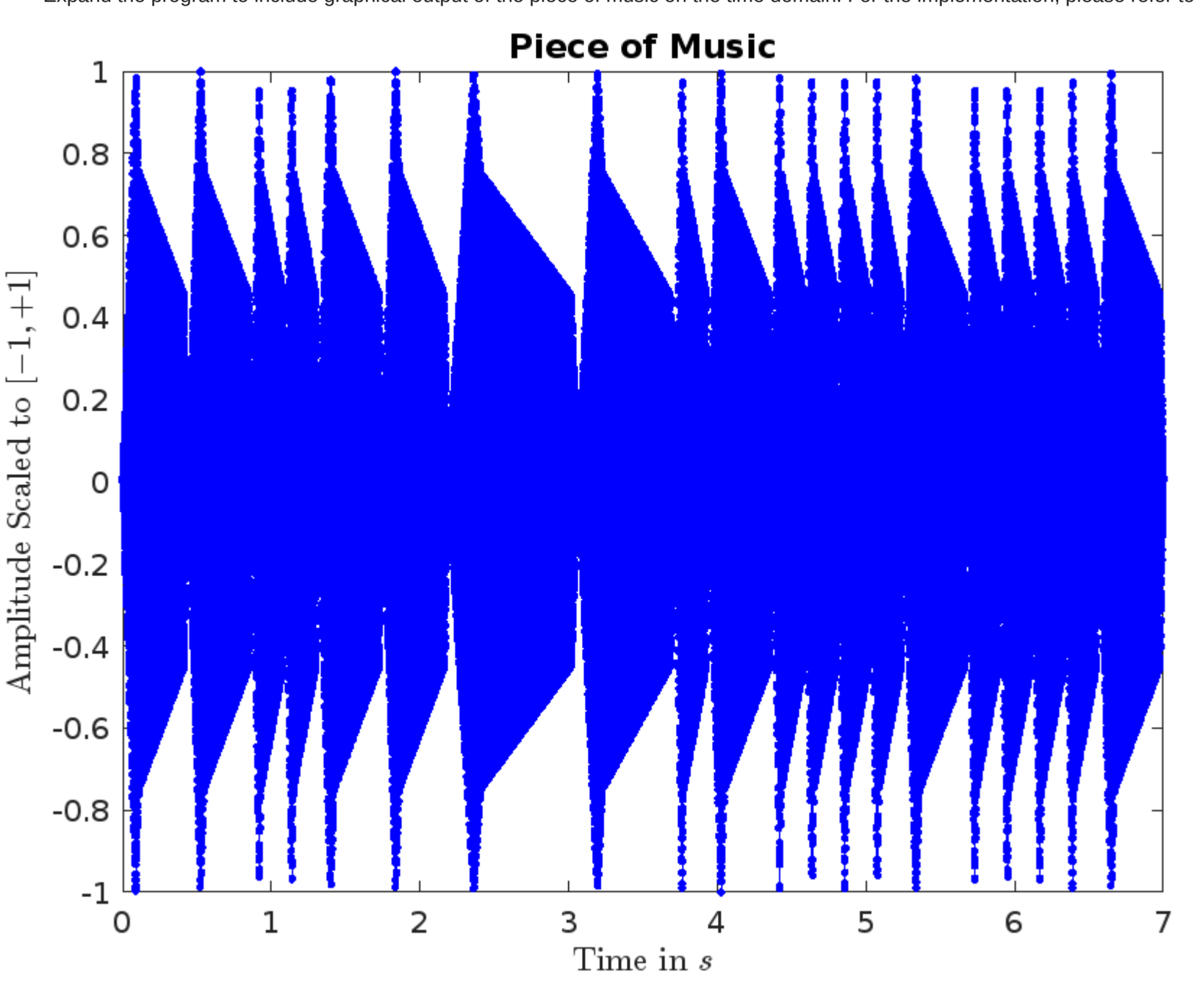


4. Task

Audio Signal Synthesis

- Familiarize yourself with the program `audiosynth.m`. This is achieved by running the default script without setting any parameters. The resulting audio of the pure sine wave is pretty underwhelming and requires additional tweaking.

- Expand the program to include graphical output of the piece of music on the time domain. For the implementation, please refer to `audiosynth.m`. The final plot of the piece is shown below, which looks indeed convincing, though a little cluttered:

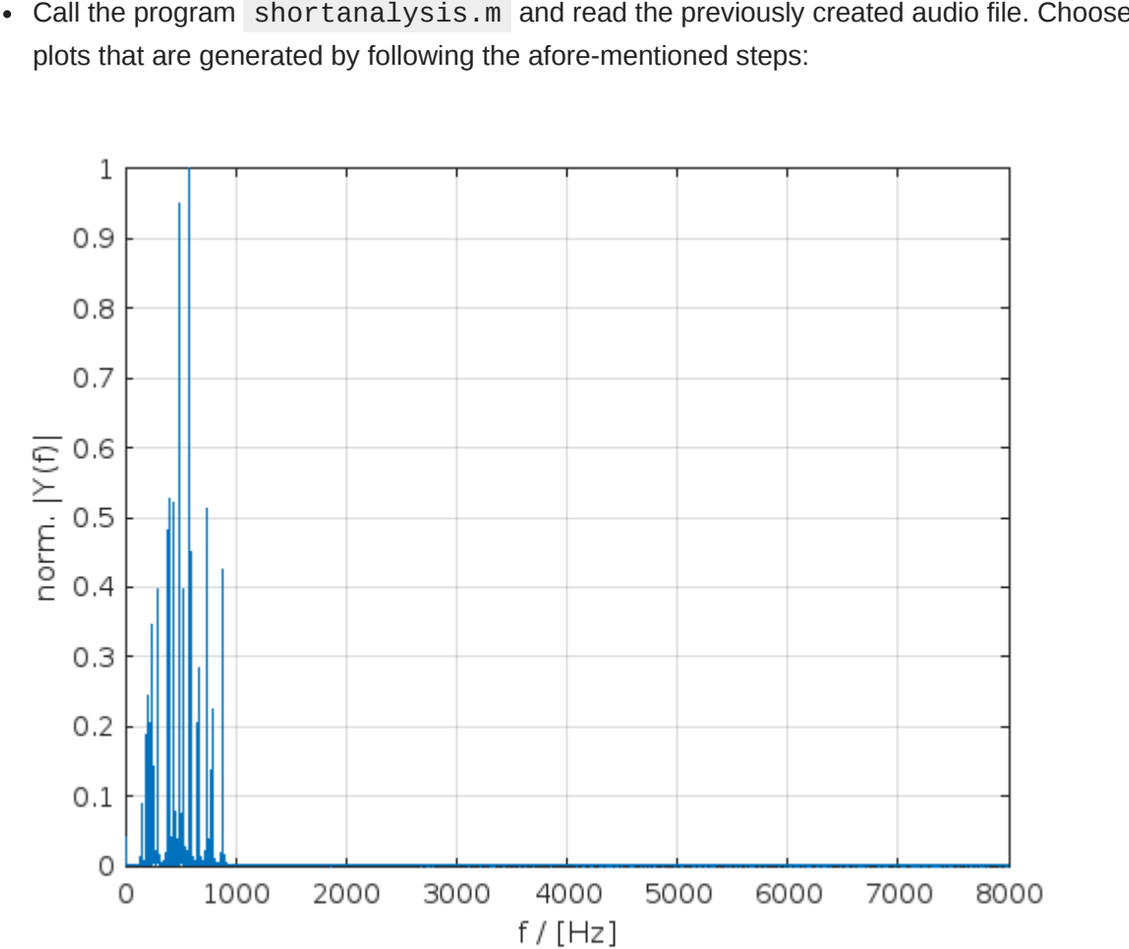


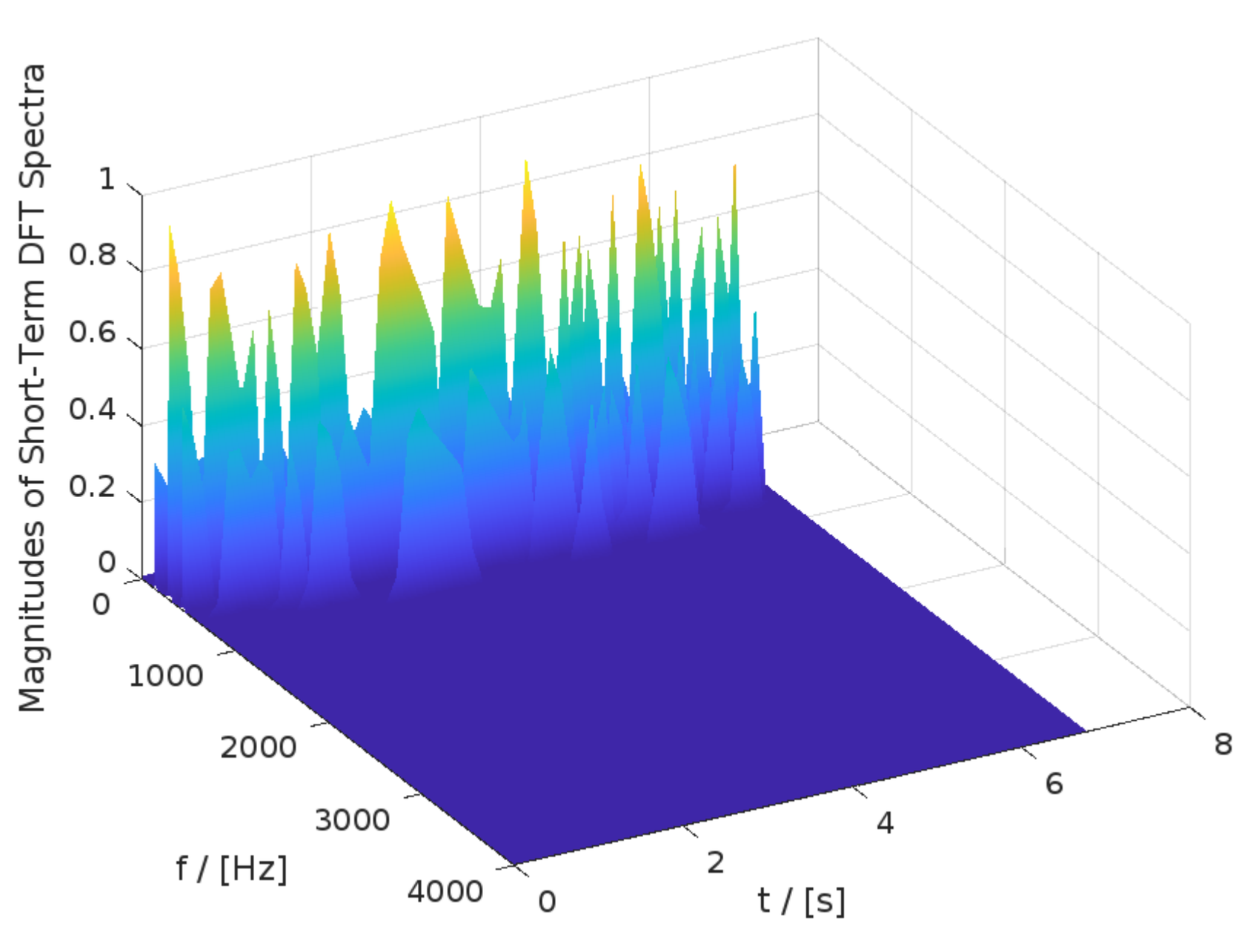
- Try different settings for the sampling frequency f_s, the time scale T as well as for the parameters of the ADSR profile. After some trial and error, the following settings and parameters are chosen: $f_s = 16$ kHz (so-called "wideband frequency"), $T = 1.75$ s, $t_A = 0.20$, $t_D = 0.25$, $t_S = 0.95$, $E_D = 0.75$ and $E_S = 0.45$. Though this configuration does not deviate strongly from the initial values, it already allows for smooth transitions between notes and thus leads to a rather natural listening experience. For details, please see the edited versions of `audiosynth.m` as well as `adsr_profile.m`.

- Further enhance the sound impression by adding harmonics. To that end, the second and third harmonics are added to the original signal before applying the ADSR profile. The harmonics are weighted by 2.75 and 1.50, respectively, placing greater emphasis on the second harmonic. This indeed significantly improves the audio quality. Again, the concrete implementation can be inspected in `audiosynth.m`.

- Create a WAV file from your final piece of music. Using MATLAB's `audiowrite` function, a WAV file is generated and saved under the name `assignment5_4.wav`. The audio file is inlcuded in the submission and can be listened to.

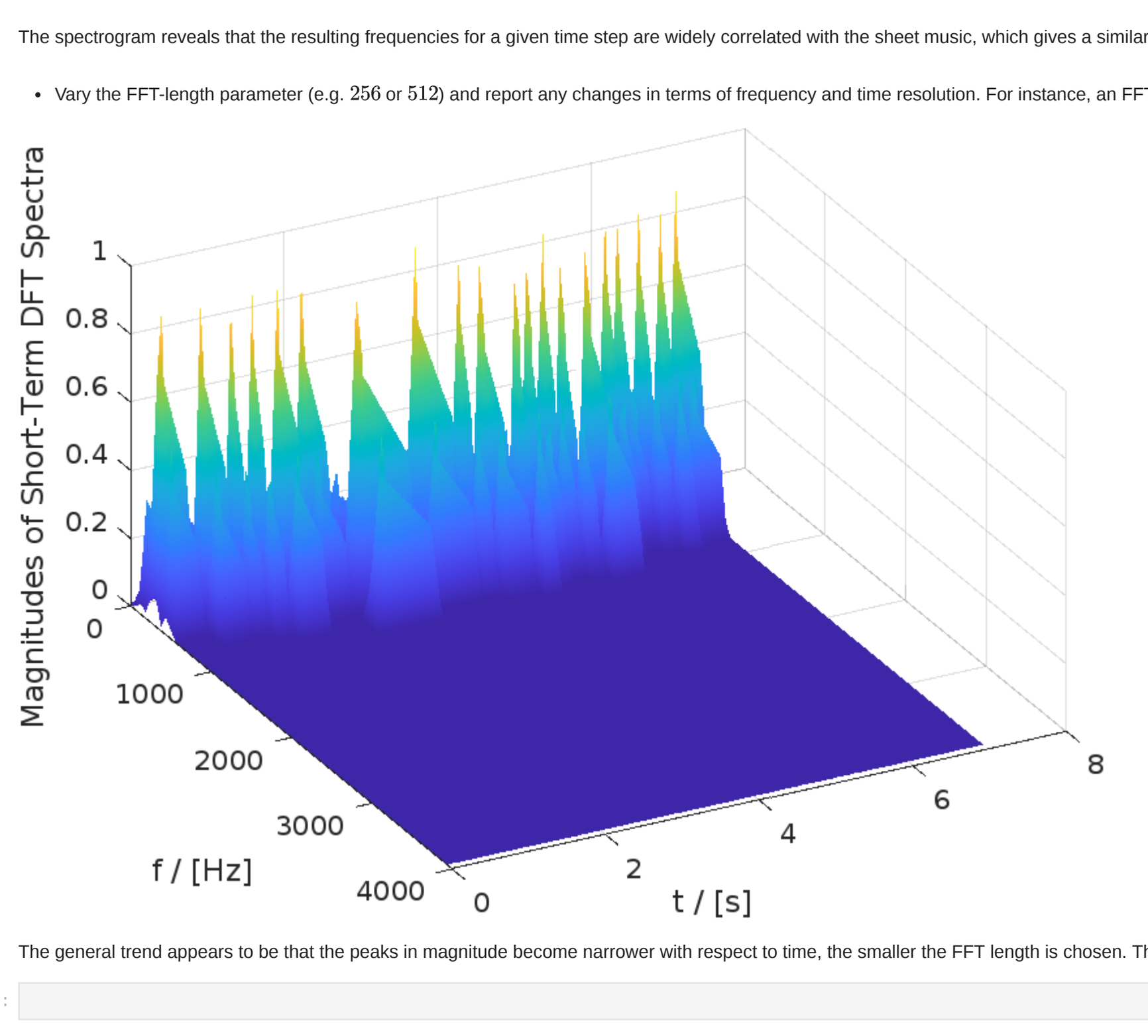
- Call the program `shortanalysis.m` and read the previously created audio file. Choose an FFT length of $M = 4096$ and an overlap of $OL = 2$. Rotate the spectrogram to observe the frequency content over time and compare it to the sheet music. Find below the plots that are generated by following the afore-mentioned steps:





The spectrogram reveals that the resulting frequencies for a given time step are widely correlated with the sheet music, which gives a similar picture considering the peak values in magnitude.

- Vary the FFT length (e.g. 256 vs 512) and report any change in terms of frequency and time resolution. For instance, an FFT length of $M = 256$ leads to the following spectrogram:



The general trend appears to be that the peaks in magnitude become narrower with respect to time, the smaller the FFT length is chosen. The frequencies, however, seem largely unchanged.