Norwegian University of Science and Technology
Department of Electronics and Telecommunications

TTT4120 Digital Signal Processing - Suggested solution for problem set 6 (Python)

In [1]:
```
import numpy as np
from matplotlib import pyplot as plt
from scipy import signal
```

Problem 1

(a).

The DTFT of $x(n)$ can be written as

$$X(\omega) = \sum_{n=0}^{N_x-1} 0.9^n e^{-j\omega n} = \sum_{n=0}^{N_x-1} (0.9e^{-j\omega})^n = \frac{1-(0.9e^{-j\omega})^{N_x}}{1-0.9e^{-j\omega}},$$

where we have used the expression for the sum of a geometric series.

Since $\omega = 2\pi f$, we obtain $X(f)$ as

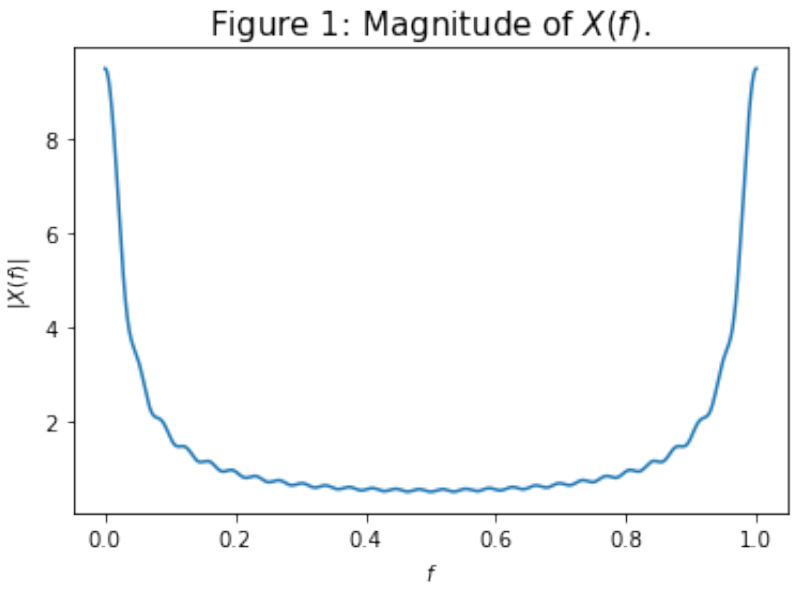
$$X(f) = \frac{1-(0.9e^{-j2\pi f})^{N_x}}{1-0.9e^{-j2\pi f}}$$

This function can be plotted in python by the following commands.

In [2]:

```
Nx = 28
alpha = 0.9

freq_axis = np.linspace(0, 1, 1000)
X_f = ((1-(alpha*np.exp(-1j*2*np.pi*freq_axis))**Nx)/(1-alpha*np.exp(1j*2*
plt.plot(freq_axis, np.abs(X_f))
plt.title('Figure 1: Magnitude of $X(f)$.', fontsize = 15)
plt.ylabel('$|X(f)|$')
plt.xlabel('$f$')
plt.show()
```



(b).

The following lines will generate the signal x(n) and compute its DFTs.

In [3]:

```
xn_x_axis = np.linspace(0, Nx-1, Nx)
xn = alpha**xn_x_axis

Nx_vec = [Nx*2, Nx, int(Nx/2), int(Nx/4)]

for i in range(4):
    Nxs = Nx_vec[i]
    xk = np.fft.fft(xn, n = Nxs)
```

(c).

The relationship between f and k is given by

$$f = \frac{k}{N},\ k = 0, \ldots, N-1$$

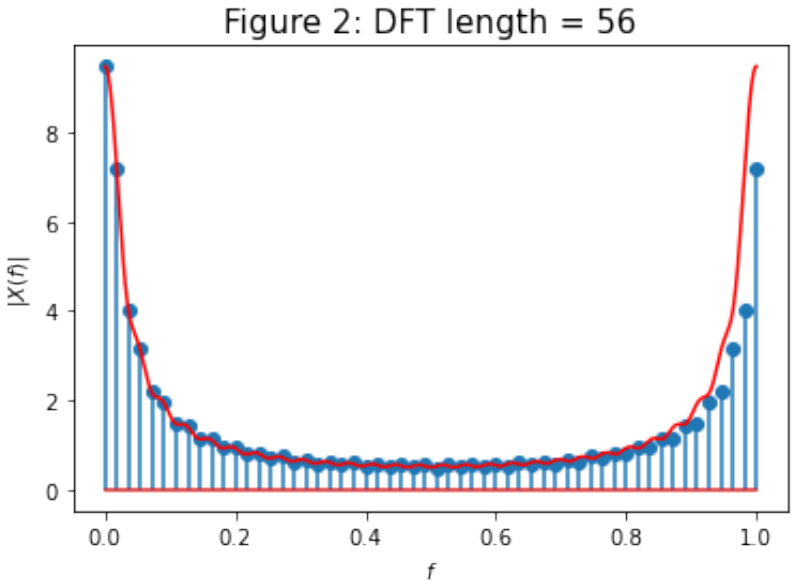
where N is the length of the DFT.

(d).

Code for plotting the magnitude the DFTs together with the DTFT is as follows.

In [4]:

```
for i in range(4):
    Nxs = Nx_vec[i]
    xk = np.fft.fft(xn, n = Nxs)
    plt.stem(np.linspace(0, 1, Nxs), np.abs(xk))
    plt.plot(freq_axis, np.abs(X_f), 'r')
    plt.title('Figure ' + str(2+i) + ': DFT length = ' + str(Nxs), fontsize
    plt.xlabel('$f$')
    plt.ylabel('$|X(f)|$')
    plt.show()
```



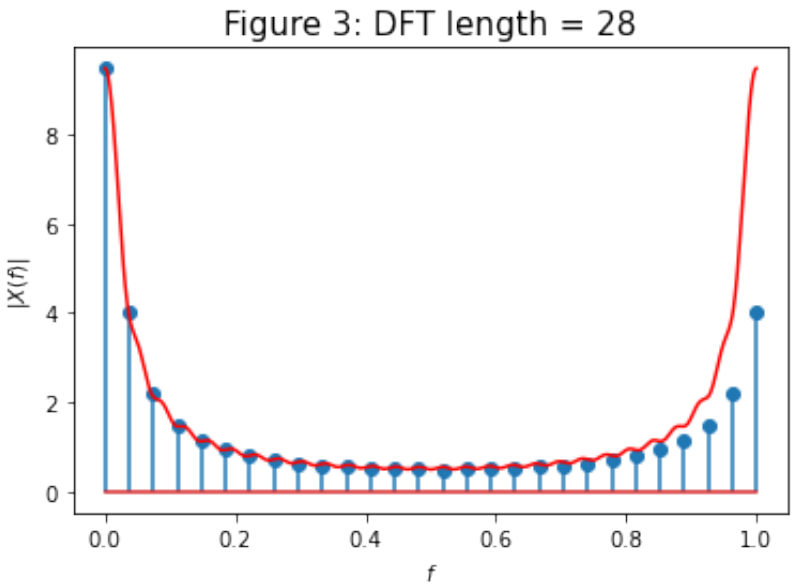


Figure 4: DFT length = 14

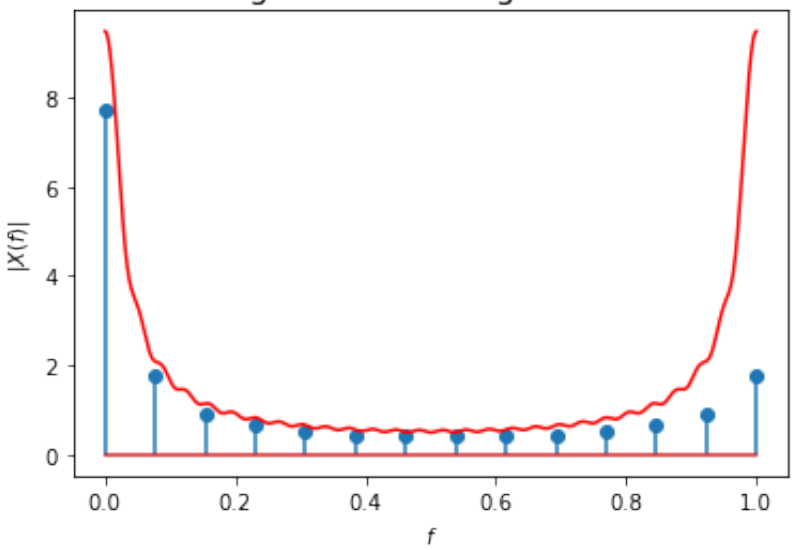
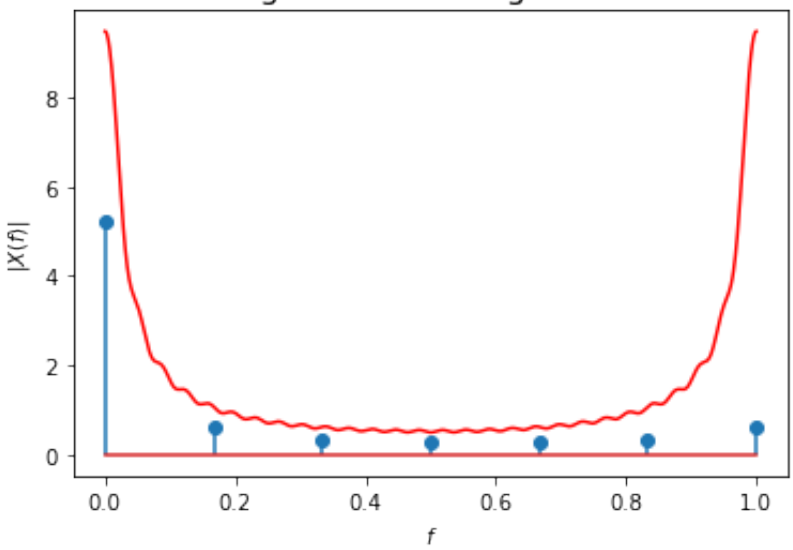


Figure 5: DFT length = 7



This results in the plot shown in figure 2-5. We see that if the DFT-length is less than N_x, the DFT-values will not represent exact the samples of the DTFT.

(e).

It follows from the symmetry and periodicity properties of the DTFT and DFT that if their values are known for $f \in [0, 0.5]$, they would be known for all values of f.

Problem 2

(a).

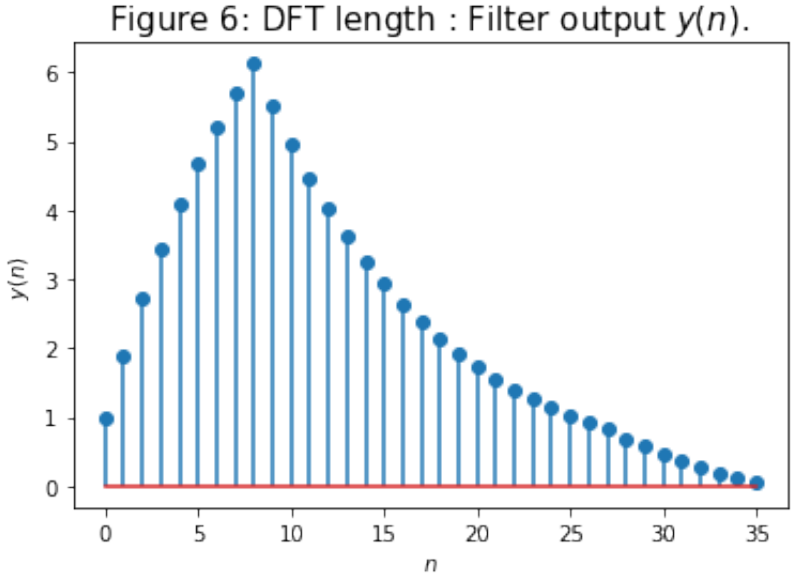
The following code compute and plot the output signal $y(n)$. The resulting plot is shown in figure 6.

In [5]:

```
Nx = 28
Nh = 9
alpha = 0.9

hn = np.ones(Nh)
xn_x_axis = np.linspace(0, Nx-1, Nx)
xn = alpha**xn_x_axis

yn = np.convolve(hn, xn)
plt.stem(yn)
plt.title('Figure 6: DFT length : Filter output $y(n)$. ', fontsize = 15)
plt.xlabel('$n$')
plt.ylabel('$y(n)$')
plt.show()
```



The length of the output signal $y(n)$ is given by $Ny = Nx + Nh - 1 = 36$. This can be verified from the plot.

(b).

$y(n)$ can be found by the algorithm given below. The resulting signal $y(n)$ for the three different choices of DFT/IDFT lengths are shown in figure 7-10.

In [6]:

```
Ny = int(len(yn))
Nys = [Ny*2, Ny, int(Ny/2), int(Ny/4)]

for i in range(4):
    N = Nys[i]
    xk = np.fft.fft(xn, n = N)
    hk = np.fft.fft(hn, n = N)
    yk = xk*hk
    yn0 = np.fft.ifft(yk, n = int(N))
    plt.stem(yn0.real)
    plt.xlabel('$n$')
    plt.ylabel('$y(n)$')
    plt.title('Figure ' + str(7+i) + ': DFT length = ' + str(N), fontsize =
    plt.show()
```

Figure 7: DFT length = 72

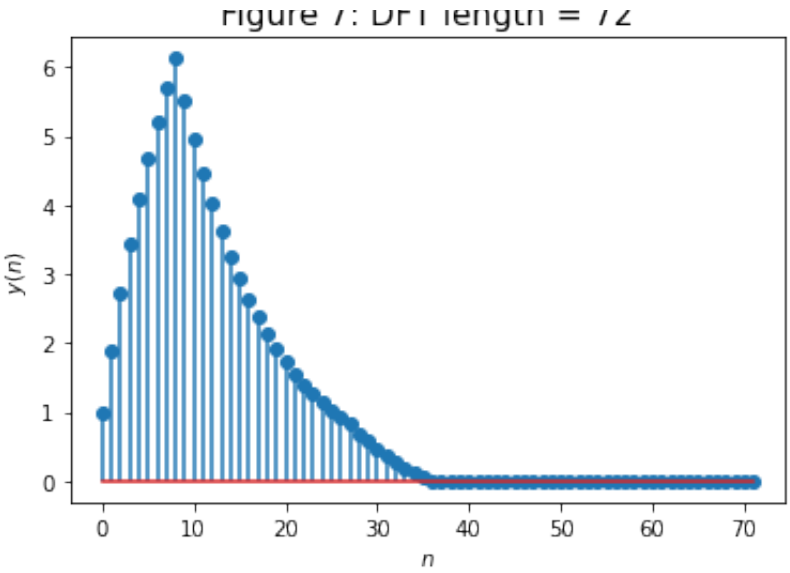


Figure 8: DFT length = 36

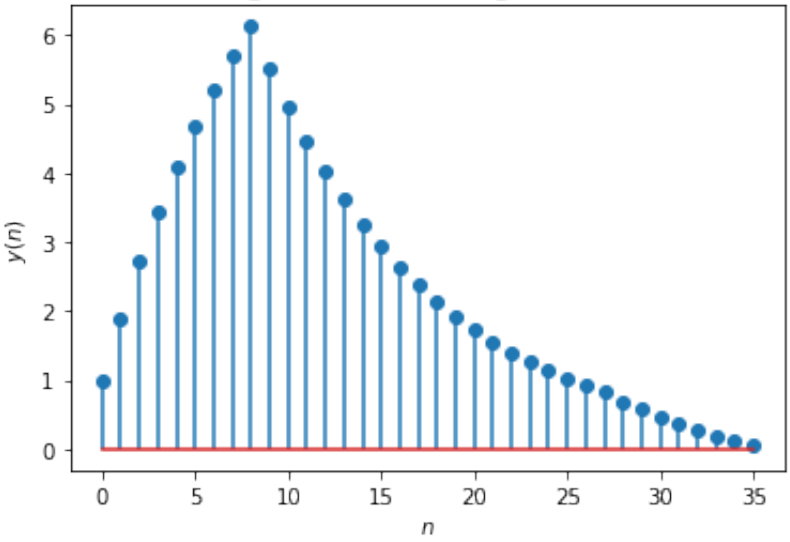
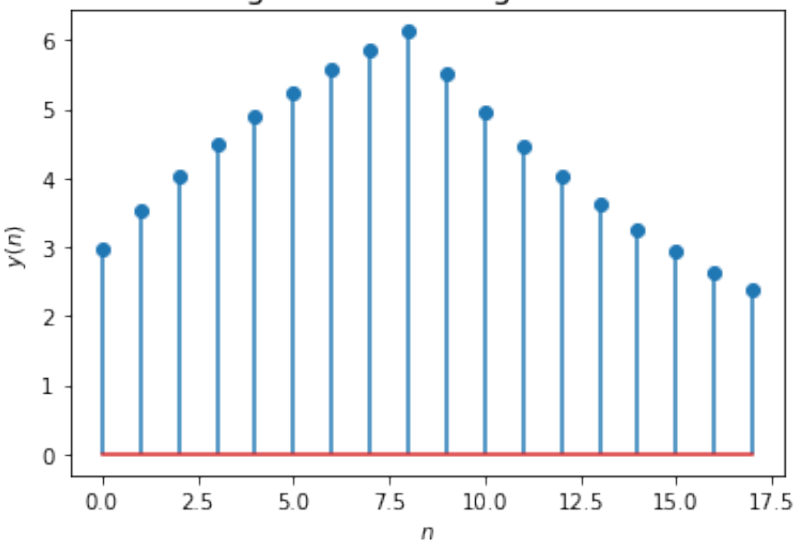
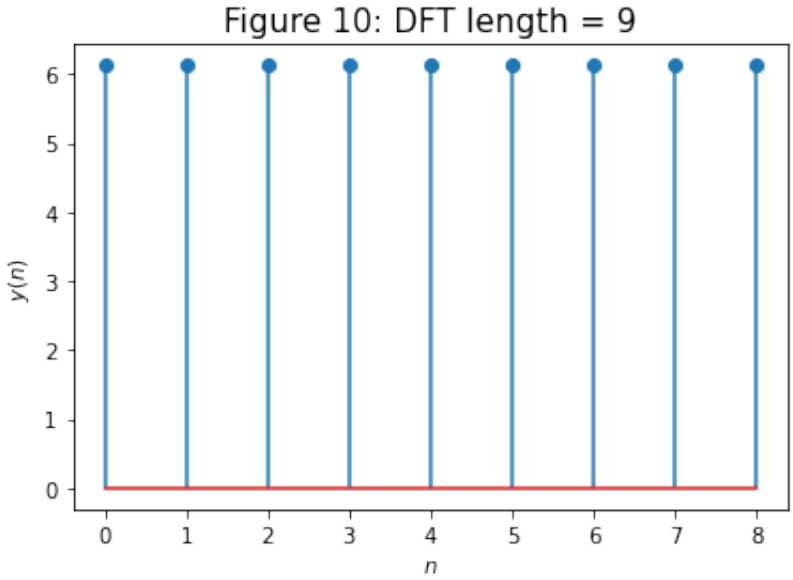


Figure 9: DFT length = 18





We know from theory that the length of the DFT must be at least equal to Ny to avoid time domain aliasing. This is verified by the plots.

Note also that the output signal computed using DFT length $N = \frac{Ny}{2} = 18$ is a given as one period of a periodic extension of $y(n)$ with period $\frac{Ny}{2} = 18$. This is easily verified if the signal is plotted in the same plot as the original $y(n)$ (use different colours).

Problem 3

(a).

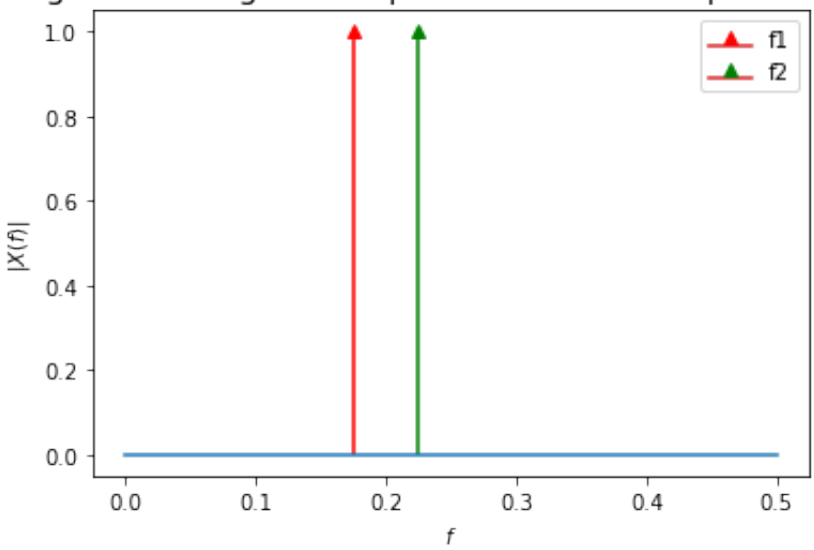
Figure 11 shows the magnitude spectrum of the sampled signal on the interval $f \in [0, 0.5]$.

In [7]:

```
f1 = 7/40
f2 = 9/40

plt.stem([f1], [1], 'r', markerfmt = '^r',  label = 'f1')
plt.stem([f2], [1], 'g', markerfmt = '^g', label = 'f2')
plt.legend()
plt.plot( np.linspace(0, 0.5, 100), np.zeros(100))
plt.xlabel('$f$')
plt.ylabel('$|X(f)|$')
plt.title('Figure 11: Magnitude spectrum of the sampled signal', fontsize =
plt.show()
```

Figure 11: Magnitude spectrum of the sampled signal



(b).

Code for computing a spectral estimate from a finite signal segment is shown below. Spectral magnitudes estimated using the different segment lengths are shown in figure 12-15.

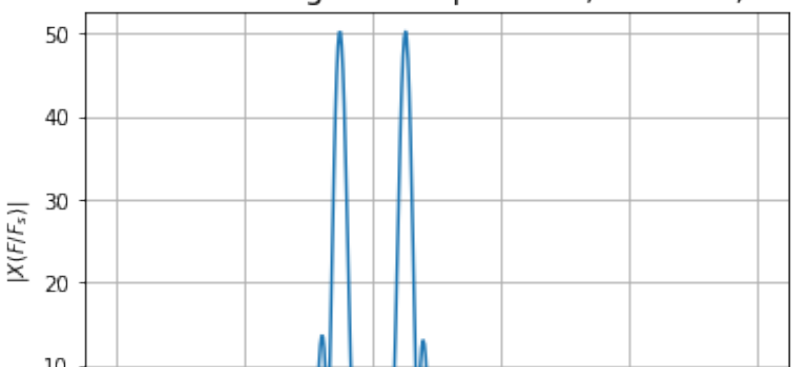
In [8]:

```
Nx = 100
N_dft = 1024

N_vec = [Nx, 1000, 30, 10]
figure_num = 11

for N in N_vec:
    xn = []
    n_vec = np.linspace(0, N-1, N)
    xn = (np.sin(2*np.pi*n_vec*f1)+np.sin(2*np.pi*n_vec*f2))
    xk = np.fft.fft(xn, n = N_dft)
    xk = np.abs(xk)
    xk_x_axis = np.linspace(0, 1, N_dft)
    plt.plot(xk_x_axis[:int(N_dft/2)], xk[:int(N_dft/2)])
    figure_num += 1
    navn = 'Figure ' + str(figure_num) + ': Estimates of the magnitude spec
    plt.title(navn, fontsize = 15)
    plt.xlabel('$F$ [Hz]')
    plt.ylabel('$|X(F/F_s)|$')
    plt.grid()
    plt.show()
```

Figure 12: Estimates of the magnitude spectrum, N = 100, DFT lenght = 1024



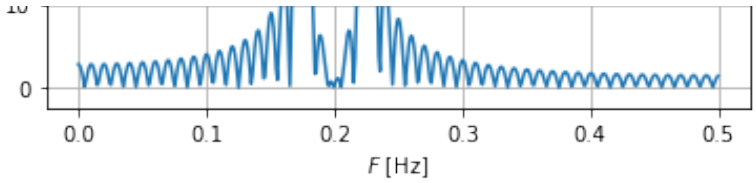


Figure 13: Estimates of the magnitude spectrum, N = 1000, DFT lenght = 1024

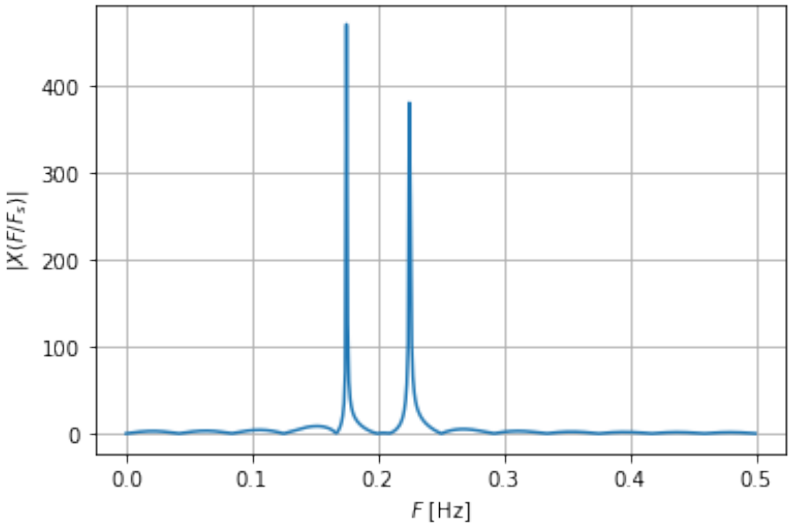


Figure 14: Estimates of the magnitude spectrum, N = 30, DFT lenght = 1024

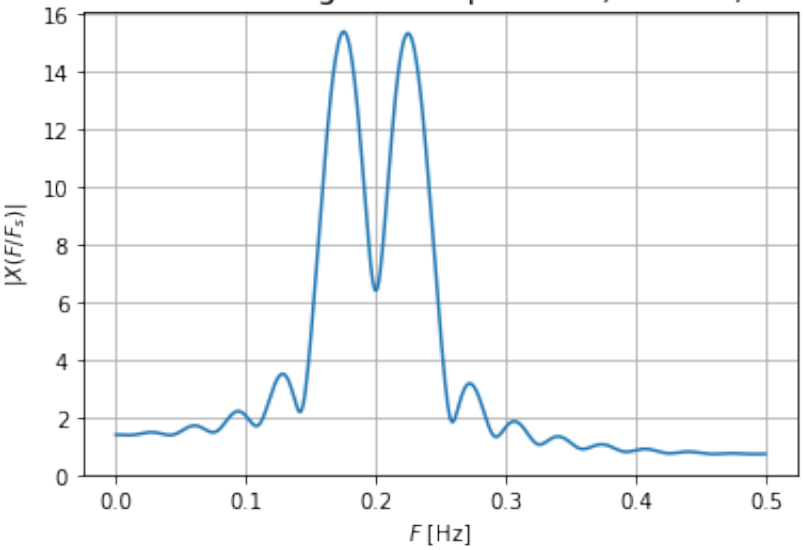
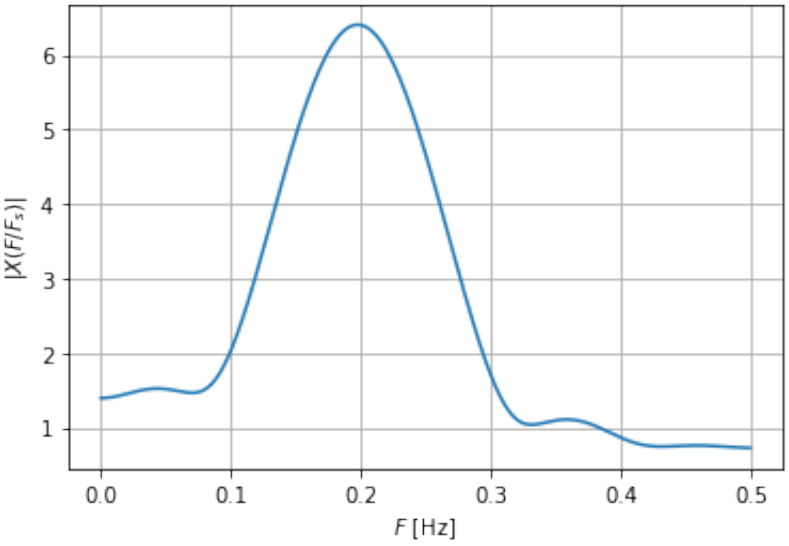


Figure 15: Estimates of the magnitude spectrum, N = 10, DFT lenght = 1024



We observe that shorter segments result in less accurate spectral estimates. The effect of spectral leakage is evident, as the frequency content is no longer limited to only two spectral components. The spectral resolution becomes poorer as the segment length decreases, and for the segment length of 10 samples, it is no longer possible to distinguish the two original frequency components.

(c).

Spectral magnitudes estimated using segment lengths 100 and different DFT lengths are shown in figure 16-18.

In [9]:

```
N_dft_vec = [1024, 256, 128]
N = 100

for N_dft in N_dft_vec:
    xn = []
    n_vec = np.linspace(0, N-1, N)
    xn = (np.sin(2*np.pi*n_vec*f1)+np.sin(2*np.pi*n_vec*f2))
    xk = np.fft.fft(xn, n = N_dft)
    xk = np.abs(xk)
    xk_x_axis = np.linspace(0, 1, N_dft)
    plt.plot(xk_x_axis[:int(N_dft/2)], xk[:int(N_dft/2)])
    figure_num += 1
    navn = 'Figure ' + str(figure_num) + ': Estimates of the magnitude spec
    plt.title(navn, fontsize = 15)
    plt.xlabel('$F$ [Hz]')
    plt.ylabel('$|X(F/F_s)|$')
    plt.grid()
    plt.show()
```

Figure 16: Estimates of the magnitude spectrum, N = 100, DFT lenght = 1024

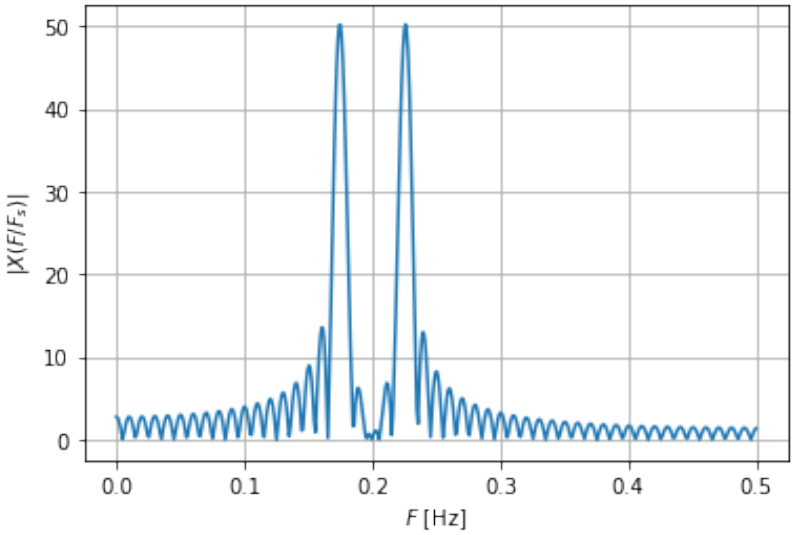


Figure 17: Estimates of the magnitude spectrum, N = 100, DFT lenght = 256

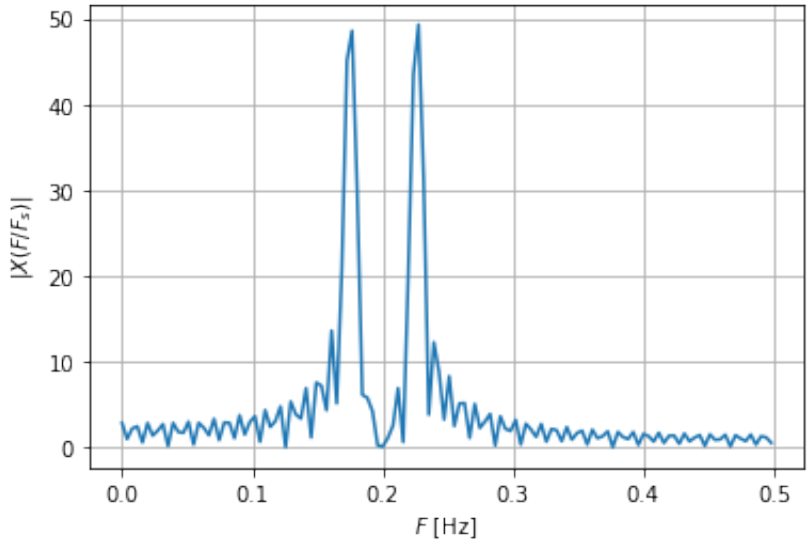
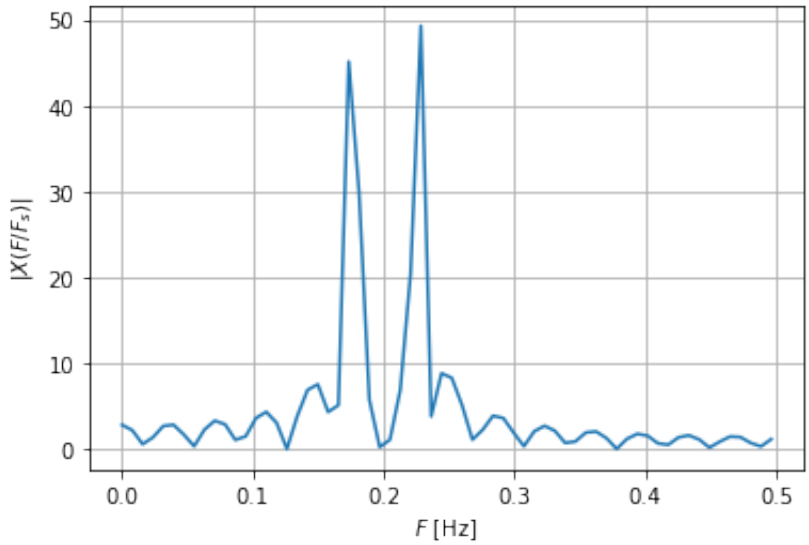


Figure 18: Estimates of the magnitude spectrum, N = 100, DFT lenght = 128



Denoting the true spectrum as $H(\omega)$, the length L of the signal segment will determine the accuracy of the spectral estimate $\hat{H}(\omega)$, while the transform length N will determine spacing between the samples of $\hat{H}(\omega)$. (The DFT must be longer then the segment length to avoid aliasing).

Problem 4

(a).

Fast Fourier Transform (FFT) is the name of some efficient computational algorithms for computing the DFT.

(b).

Radix-2 FFT algorithm is an efficient algorithm for DFT computation which can be applied when $N = 2^v$, where ν is an integer (zero padding might have to be used to extend the number of data points to N).

The idea of the algorithm is to devide the sequence $x(n)$ into two sequences $f_1(n)$ and $f_2(n)$ of length $\frac{N}{2}$, compute their DFTs and combine them to obtain $X(k)$.

It can be shown that the number of operations needed to compute $X(k)$ in this way is approximately two times smaller then in the case of direct computation of $X(k)$ for large N. This is due to the periodicity and symmetry properties of W_N^{kn}.

This fact is used iteratively for computation of DFTs of $f_1(n)$ and $f_2(n)$, so that the direct DFT computation has to be applied only to subsequencies of length 2.

(c).

From the formula for DFT we can observe that for each value of k, direct computation of $X(k)$ needs N complex multiplications and $N - 1$ complex additions. To compute all N values of the DFT then requires N^2 complex multiplications and $N^2 - N$ complex additions.

(d).

$$X(k) = \sum_{n=0}^{N-1} x(n) W_N^{kn}, k = 0, 1, \ldots, N-1$$
$$= \sum_{\text{n even}} x(n) W_N^{kn} + \sum_{\text{n odd}} x(n) W_N^{kn}$$
$$= \sum_{m=0}^{(N/2)-1} x(2m) W_N^{2mk} + \sum_{m=0}^{(N/2)-1} x(2m+1) W_N^{k(2m+1)}$$

Substitute $W^2*N* = *W*{N/2}. We can then write:

$$X(k) = \sum_{m=0}^{(N/2)-1} f_1(m) W_{N/2}^{mk} + W_N^k \sum_{m=0}^{(N/2)-1} f_2(m) W_{N/2}^{mk}$$
$$= F_1(k) + W_N^k F_2(k), k = 0, 1, \ldots, N-1$$

$F_1(k)$ and $F_2(k)$ are both periodic with period $N/2$. That is, we have $F_1(k + N/2) = F_1(k)$ and $F_2(k + N/2) = F_2(k)$. We also have $W_N^{k+N/2} = -W_N^k$. We can write:

$$X(k) = F_1(k) + W_N^k F_2(k), k = 0, 1, \ldots, N/2 - 1$$
$$X(k + N/2) = F_1(k) - W_N^k F_2(k), k = 0, 1, \ldots, N/2 - 1$$

(e).

The direct computations of $F_1(k)$ and $F_2(k)$ both requires $(N/2)^2$ complex multiplications. The are also $N/2$ complex multiplications in computing $W_N^k F_2(k)$. The number of complex multiplications required to compute $X(k)$ is $2(N/2)^2 + N/2 = N^2/2 + N/2$.

Comparing with the results in (c) we can see that for large N the reduction in number of multiplications are about a factor of 2.