Σκοπός αυτής της άσκησης είναι να εξοικειωθείτε με τις έννοιες και τις λειτουργίες που σχετίζονται με το FFT και το iFFT.

1. Στις ηχογραφήσεις που κάνατε στην προηγούμενη άσκηση, εφαρμόστε το FFT και δείτε αν οι συχνότητες που παίρνετε συμφωνούν με τις εκτιμήσεις που κάνατε "με το μάτι" για τη συχνότητα του σήματος πάνω στην κυματομορφή.
2. Δημιουργήστε ένα τεχνητό σήμα ως άθροισμα δύο ημιτόνων συχνοτήτων 50 και 70 Hz, θεωρώντας μια συχνότητα δειγματοληψίας 44100.
   A. Ποιο είναι το ελάχιστο μέγεθος παραθύρου (σε δύναμη του 2) που χρειάζεται να εφαρμόστε στο FFT για να έχετε τις δύο αυτές συχνότητας σε διαφορετικά "κουτάκια" FFT (FFT bins);
   B. Αν ήταν μισή η συχνότητα δειγματοληψίας, ποιο θα ήταν το ελάχιστο αυτό μέγεθος;
3. Χρησιμοποιήστε μια ηχογράφηση της επιλογής σας και εφαρμόστε ένα φασματικό φίλτρο για να την παραμορφώσετε όπως επιθυμείτε.

Προθεσμία: Τρίτη 2 Δεκ. 2020 βράδυ.

In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
import scipy.io.wavfile as wf
import matplotlib.patches as mpatches
import matplotlib.pyplot as plt
import sounddevice as sd
```

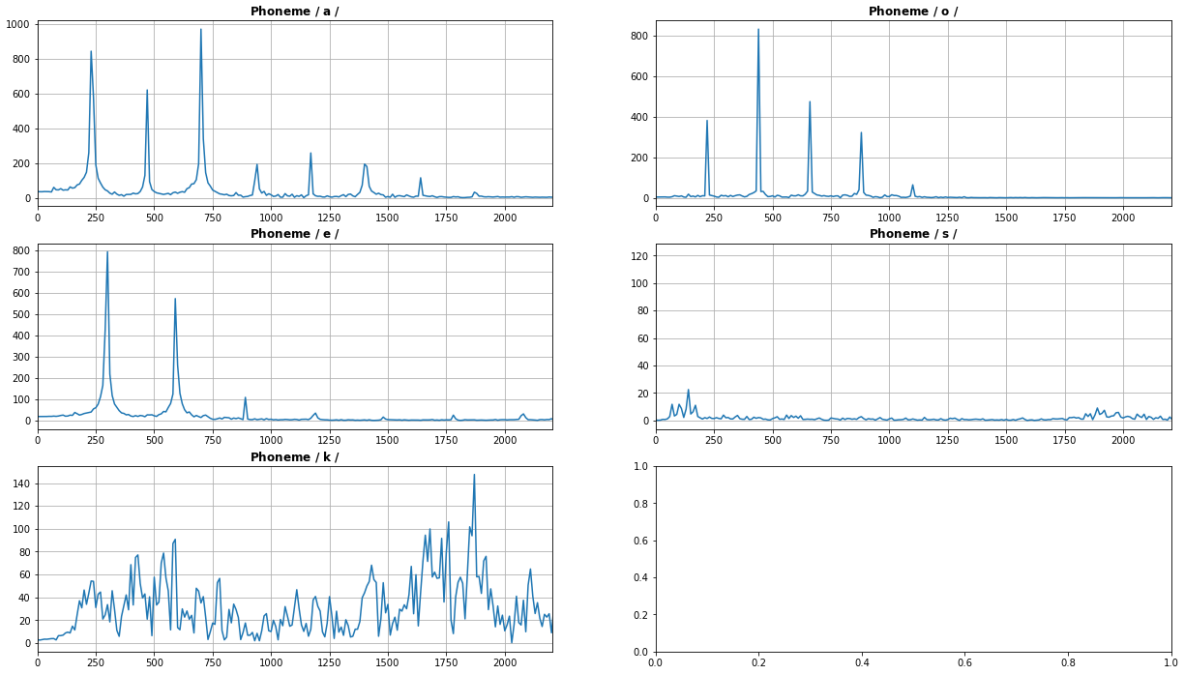
Task 1

In [98]:

```
sr_a, a = wf.read('./data/a_audio.wav')
sr_o, o = wf.read('./data/o_audio.wav')
sr_e, e = wf.read('./data/e_audio.wav')
sr_s, s = wf.read('./data/s_audio.wav')
sr_k, k = wf.read('./data/k_audio.wav')

fs = 44100
power_of_2 = 10
# time_steps = 2**power_of_2
time_steps = 4410

fft_a = np.fft.fft( a[:time_steps,0] )/len( a[:time_steps,0])
fft_o = np.fft.fft( o[:time_steps,0] )/len( o[:time_steps,0])
fft_e = np.fft.fft( e[:time_steps,0] )/len( e[:time_steps,0])
fft_s = np.fft.fft( s[:time_steps,0] )/len( s[:time_steps,0])
fft_k = np.fft.fft( k[:time_steps,0] )/len( k[:time_steps,0])

freq_bins_a = np.arange( fft_a.size )/fft_a.size*fs
freq_bins_o = np.arange( fft_o.size )/fft_o.size*fs
freq_bins_e = np.arange( fft_e.size )/fft_e.size*fs
freq_bins_s = np.arange( fft_s.size )/fft_s.size*fs
freq_bins_k = np.arange( fft_k.size )/fft_k.size*fs

print(len(freq_bins))
mag_a = np.sqrt( np.power( fft_a.real , 2 ) + np.power( fft_a.imag , 2 ) )
mag_o = np.sqrt( np.power( fft_o.real , 2 ) + np.power( fft_o.imag , 2 ) )
mag_e = np.sqrt( np.power( fft_e.real , 2 ) + np.power( fft_e.imag , 2 ) )
mag_s = np.sqrt( np.power( fft_s.real , 2 ) + np.power( fft_s.imag , 2 ) )
mag_k = np.sqrt( np.power( fft_k.real , 2 ) + np.power( fft_k.imag , 2 ) )

print(max(mag_a), max(freq_bins), len(mag_a),len(mag_a))

# plt.plot( np.arange( fft_a.size )/fft_a.size*sr_original , fft_a.real )

frame_to_plot = int(time_steps/2)

fig, plots = plt.subplots(3,2,figsize=(21,12))#, sharex=True)
# fig.text(0.05,0.5, "Amplitude", ha="center", va="center", rotation=90)
# plt.xlabel("Frequency")

plots[0,0].grid(True)
plots[0,0].title.set_text(r"$\bf{Phoneme\ /\ a\ / }$")
plots[0,0].plot( freq_bins_a[:frame_to_plot] , mag_a[:frame_to_plot])
plots[0,0].set_xlim(0,len(mag_a[:frame_to_plot]))

plots[0,1].grid(True)
plots[0,1].title.set_text(r"$\bf{Phoneme\ /\ o\ / }$")
plots[0,1].plot( freq_bins_o[:frame_to_plot] , mag_o[:frame_to_plot])
plots[0,1].set_xlim(0,len(mag_o[:frame_to_plot]))

plots[1,0].grid(True)
plots[1,0].title.set_text(r"$\bf{Phoneme\ /\ e\ / }$")
plots[1,0].plot( freq_bins_e[:frame_to_plot] , mag_e[:frame_to_plot])
plots[1,0].set_xlim(0,len(mag_e[:frame_to_plot]))

plots[1,1].grid(True)
plots[1,1].title.set_text(r"$\bf{Phoneme\ /\ s\ / }$")
plots[1,1].plot( freq_bins_s[:frame_to_plot] , mag_s[:frame_to_plot])
plots[1,1].set_xlim(0,len(mag_s[:frame_to_plot]))

plots[2,0].grid(True)
plots[2,0].title.set_text(r"$\bf{Phoneme\ /\ k\ / }$")
plots[2,0].plot( freq_bins_k[:frame_to_plot] , mag_k[:frame_to_plot])
```

```
4410
968.7868267605809 44090.0 4410 4410
```

Out[98]: (0.0, 2205.0)



Task 2A

In [77]:

```
fs = 44100
t = np.arange(fs)/fs

f1 = 50
f2 = 70

s1 = np.sin( 2*np.pi*f1*t )
s2= np.sin( 2*np.pi*f2*t )

signal = s1 + s2

power_of_2 = 10
# time_steps = 2**power_of_2
time_steps = 4410

fft_a = np.fft.fft( e[:time_steps,0] )/len( a[:time_steps,0])
freq_bins = np.arange( fft_a.size )/fft_a.size*fs
print(len(freq_bins))
# freqs_a = np.fft.fftfreq(freq_bins, step)
mag_a = np.sqrt( np.power( fft_a.real , 2 ) + np.power( fft_a.imag , 2 ) )
# i_max = max(mag_a)
# freq_max = i_max * fs / len(freq_bins)
print(max(mag_a), max(freq_bins), len(mag_a),len(mag_a))
# pos = np.where(mag_a>200)[0]
# print(freq_bins[pos])

# plt.plot( np.arange( fft_a.size )/fft_a.size*sr_original , fft_a.real )
frame_to_plot = int(time_steps/2)
plt.plot( freq_bins[:frame_to_plot] , mag_a[:frame_to_plot] )
print(type(mag_a))
# ticks=np.arange(len(mag_a[:2000]))
plt.xlim(0,len(mag_a[:frame_to_plot]))
```

Task 2B

Task 3

More material

In [ ]:

```
# -*- coding: utf-8 -*-
"""
Created on Wed Nov 11 16:45:11 2020

@author: user
"""

import numpy as np
import matplotlib.pyplot as plt
import sounddevice as sd
import librosa



fs = 44100
t = np.arange(8096)/fs
freq = 50
x = np.sin( 2*np.pi*freq*t )

w = np.hanning( 8096 )

y = np.fft.fft( x )

frequency_bins = np.arange( y.size )/y.size*fs

plt.plot( np.arange(8096) , w )
plt.plot( np.arange(8096) , x )
plt.plot( np.arange(8096) , w*x )

plt.subplot(311)
plt.plot( frequency_bins , y.real )
plt.subplot(312)
plt.plot( frequency_bins , y.imag )
mag = np.sqrt( np.power( y.real , 2 ) + np.power( y.imag , 2 ) )
plt.subplot(313)
plt.plot( frequency_bins , mag )

fs = 44100
x, sr = librosa.load('audio_files/sofianos/eipa.wav', sr=fs)

sd.play( x[:4410], sr )
plt.plot( np.arange(4410), x[:4410] )
```