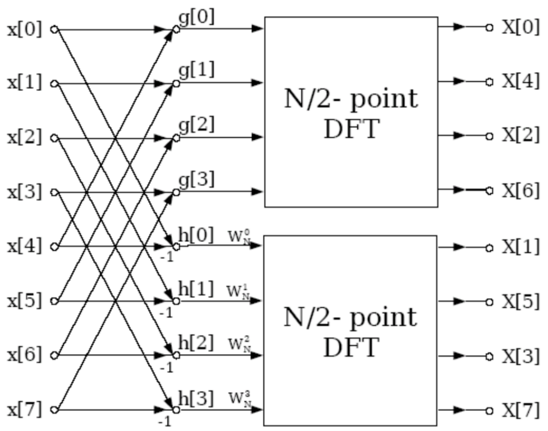
Ekin ÇELEBİ
150160152

HW4

Decimation in Frequency FFT Algorithm

N-point DFT of x[n] is, $X[k] = \sum_{n=0}^{N-1} x[n] W_N^{kn}$ $\qquad W_N = e^{-j(2\pi/N)}$ $\qquad k = 0, 1, \ldots, N-1$



DIF-FFT, input is in natural order while the output is in bit reversal order.
DIF-FFT splits the two DFTs into first half and last half of the input samples.
The radix-2 decimation in frequency algorithm arranges the discrete Fourier transform into two parts.
Each block computes even and odd indexes and places them sequentially to upper and lower parts. Each upper and lower blocks again computes DIF_FFT recursively. Therefore the number of complex multiplications are reduced from N^2 to (N/2) · log2(N)

```
def radix2(inarray,N,outarray,twid):
    '''
    inarray: audio samples
    N:number of radix points
    outarray:where to append results
    twid: to be used to calculate twiddle

    recursively apply dif fft blocks
    resuls are in bit reverse order
    '''

    xrange = int(N/2)

    if N == 1:

        outarray.append(inarray[0])
        return

    def twiddle(x,point):
        return np.exp((-2j*np.pi*x)/point)

    upper_half = np.zeros((xrange,),dtype=np.complex_)
    lower_half = np.zeros((xrange,),dtype=np.complex_)

    for i in range(0,xrange):
        upper_half[i] = inarray[i]+inarray[i+xrange]
        lower_half[i] = (inarray[i]-inarray[i+xrange])*twiddle(i,twid)


    radix2(upper_half,xrange,outarray,twid)
    radix2(lower_half,xrange,outarray,twid)
```

X->input value
Y->output value
n->number of points
For upper block:
Y[i] = X[i] + X[i+n/2]
For lower block:
Y[i] = (X[i] - X[i+n/2])*W(factor)

i=0,….N/2-1

Output array is bit reversed so it has to be shuffled again.

```
def reverse_bits(inarray,point):
    '''
    takes array and it' length as input
    reverses array to normal order
    returns output array
    '''
    outarray = np.zeros((point,),dtype=np.complex_)
    half = int(point/2)
    for i in range(0,half):
        outarray[2*i] = inarray[i]
        outarray[(2*i)+1] = inarray[i+half]
    return outarray
```

Below the demonstration of DIF-FFT Radix 2 is applied on Raw Data, LPF applied with 0dB gain and LPF applied with 5dB gain. It can be clearly seen this plot shows the frequency values present in our samples. Our FFT algorithm gives us all the frequencies in a given signal.
At times t = 10s, 20s, 30s

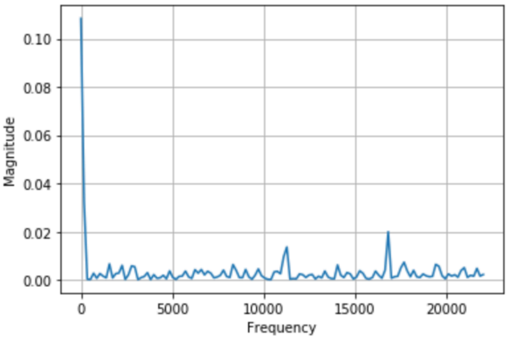
Sampling rate = 44100Hz. Due to our sampling rate we can only plot from 0 to 225000Hz according to the Nyquist sampling theorem.
To apply 256 point radix 2, only first 256 signal is taken as input.

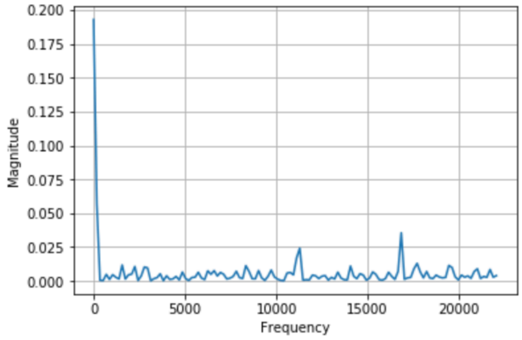
first 256 signals of a given time = (sample_rate*t)+i
i = 0,….,255

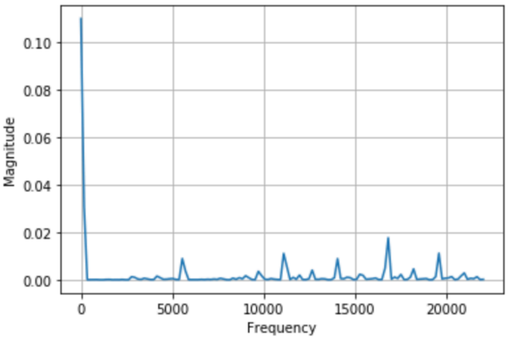
t=10s

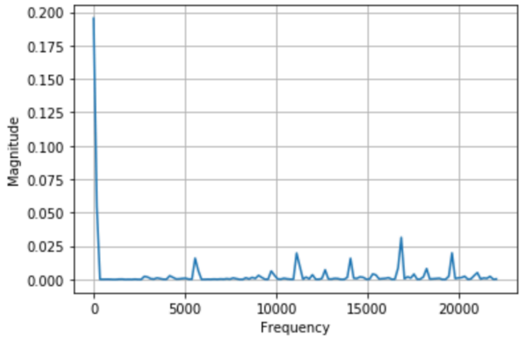
0db Gain

5db Gain





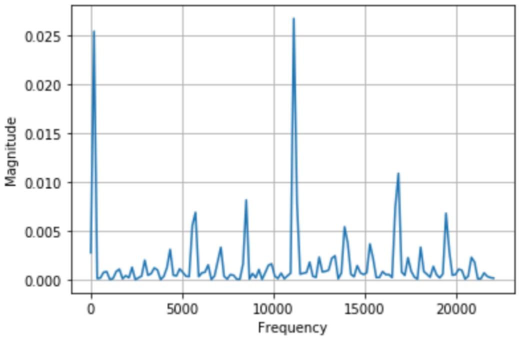


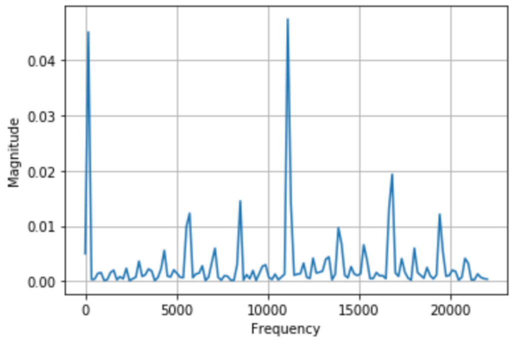


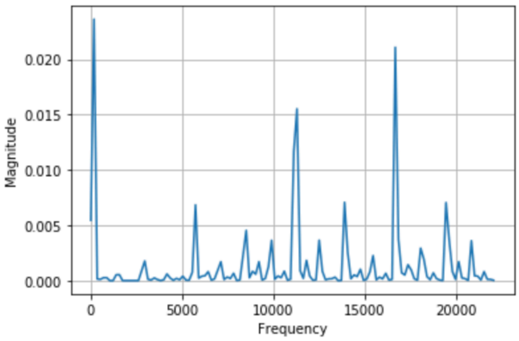
t=20s

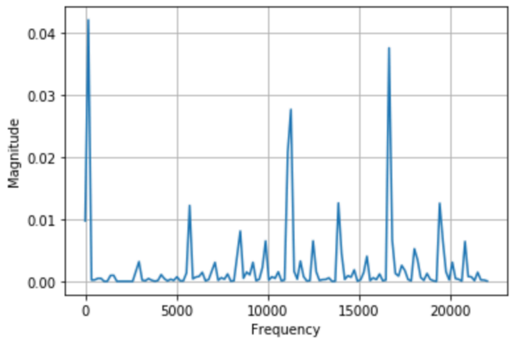
0db Gain

5db Gain



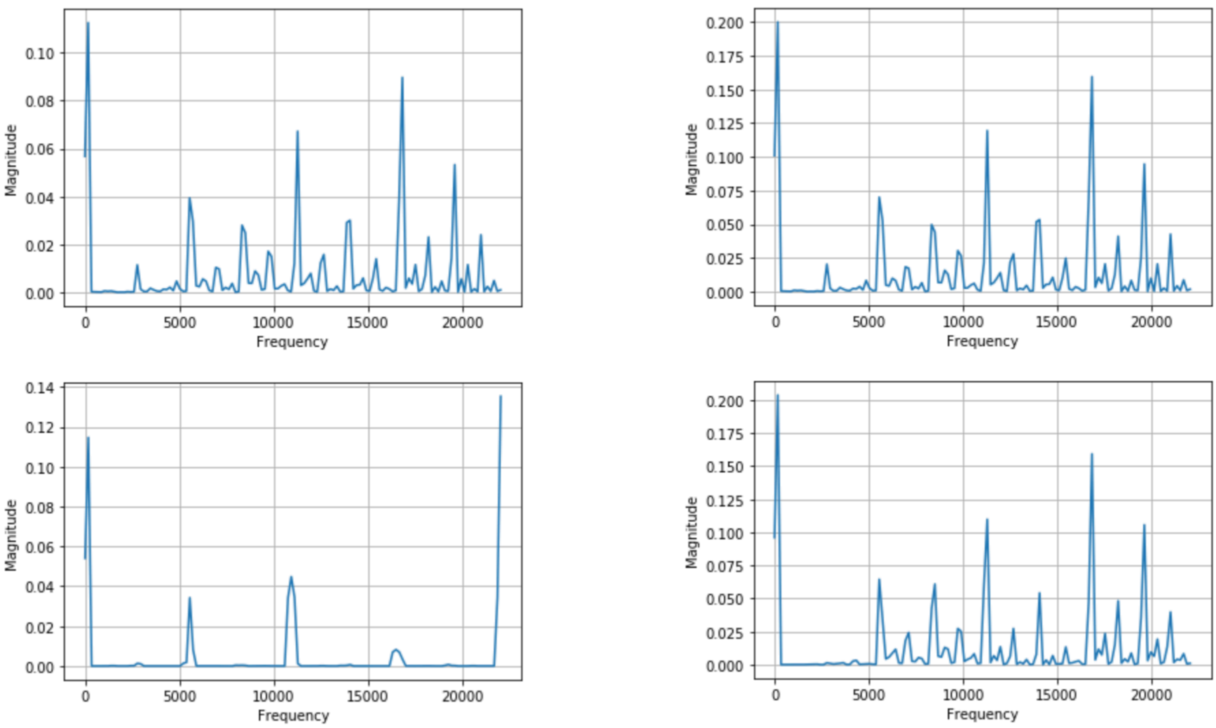






t=30s

0db Gain 5db Gain



First rows belongs to Raw Data
Second rows belongs to LPF applied data

LPF filtered out the data and with different gain factors, different magnitudes are observed. With this algorithm the signal is converted in frequency domain and the DFTs are calculated finally the results are added up.