DSAA ASSIGNMENT 2

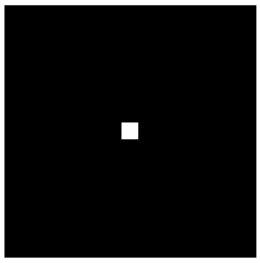
1) A recursive implementation of FFT is been done by the divide and conquer approach. Here, we instead of dividing into consecutive halves, divide the array into alternate subarrays: like 1,2,3,4 => 1,3 2,4 and so on...

The main concept used is implemented in the code as follows:

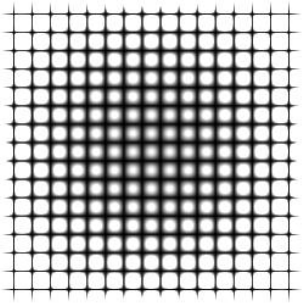
```
/*** twiddle = exp(-2*pi*1j*i/n)
     oddTerm = F_odd[i] * twiddle
     F[i] = F_even[i] + oddTerm
     F[i+n2] = F_even[i] – oddTerm ***/
```

Now, shown below are some outputs to my recursive implementation:

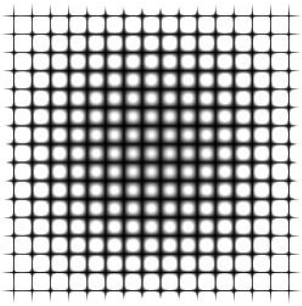
a)
input :



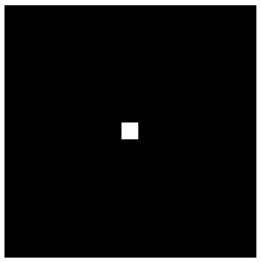
output :



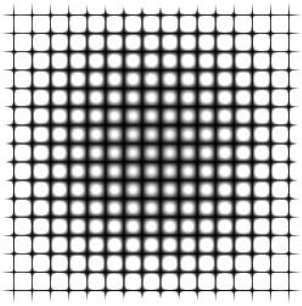
in-built output:



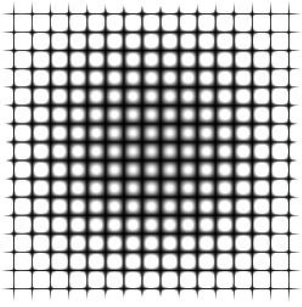
b)
input :



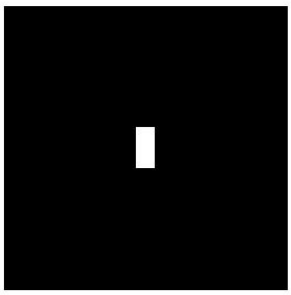
output :



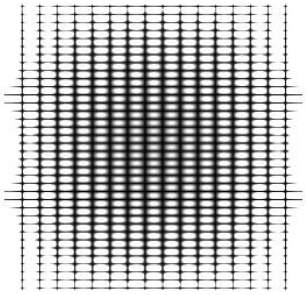
in-built output:



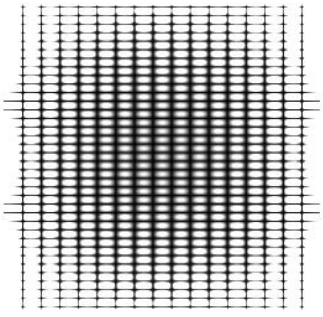
c)
input :



output :



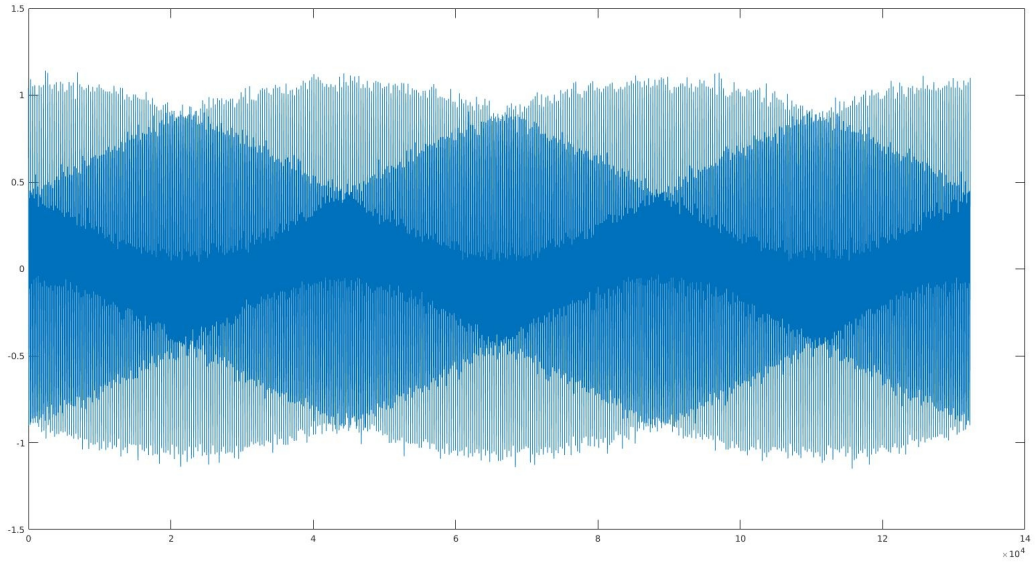
in-built output :



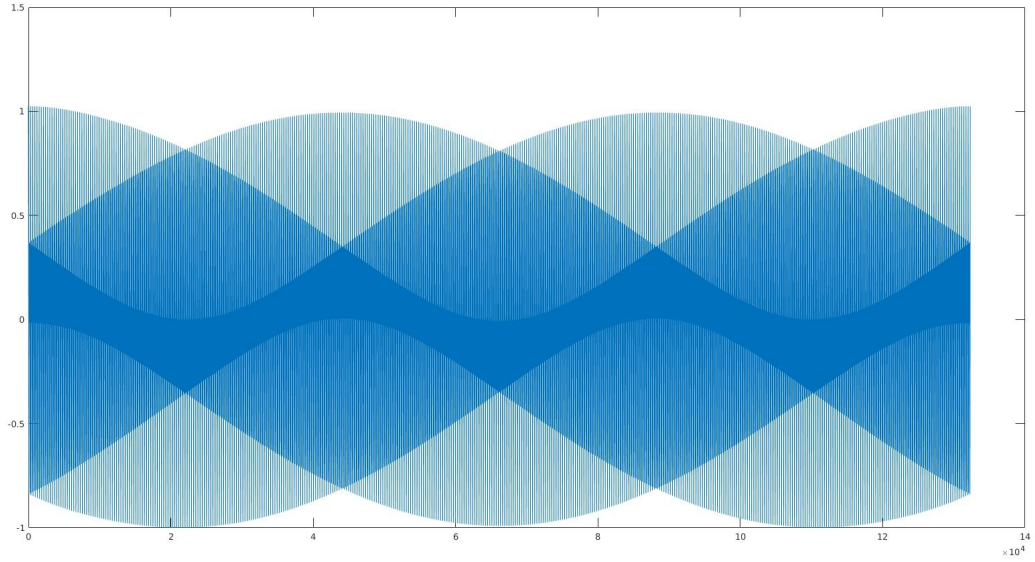
2) In the input signal, there was a significant amount of noise in the background. Firstly, I thought of merely cutting off the noise using a rect filter, but then there was a sudden break in the frequencies. Then, I decided to use a rect filter around the peaks which I found sequentially using the max function. I used a gaussian window of length 20 (probably) over the rect

filter to smoothen the transition between the noise and the lead signal, it gives an amazing output which better clarity and almost minimal noise in the background.

Noise-signal:



Denoised-signal:



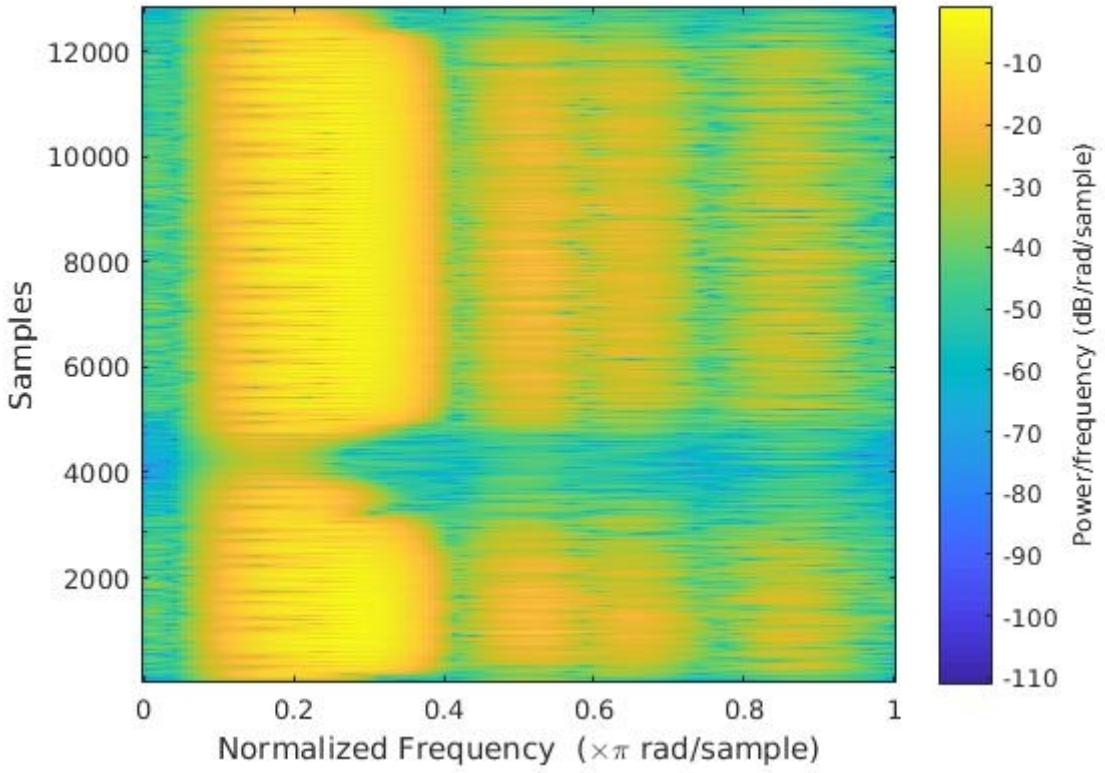
Note: For further sight into the calculations, head to the code

which is almost self-explanatory.

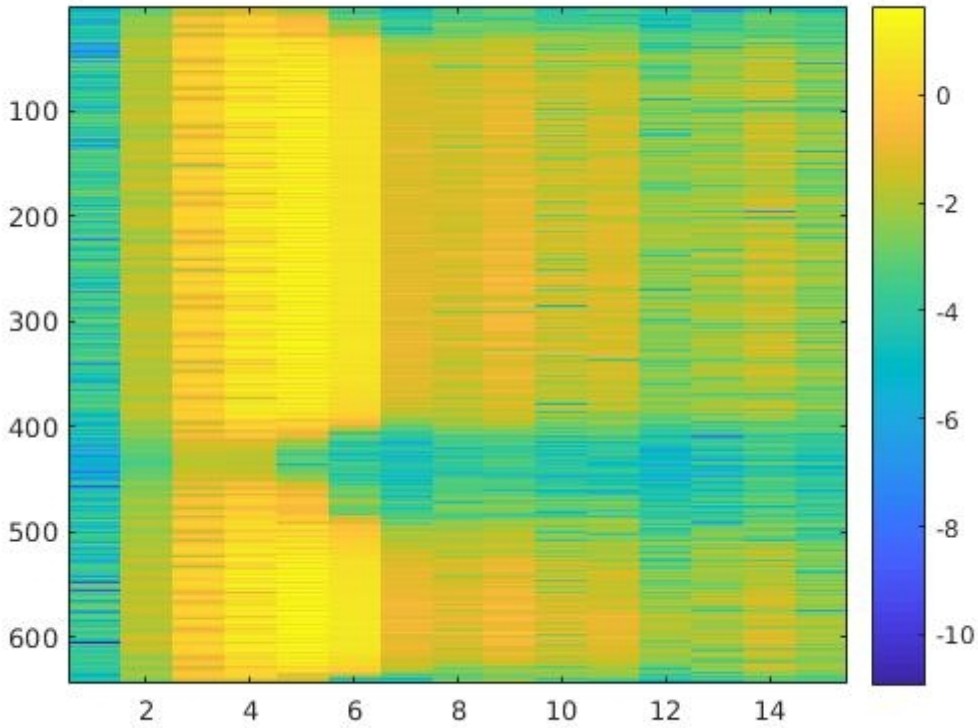
3) Let's check how similar are the spectrograms for my code and the in built function.

Train:

Inbuilt spectrogram:

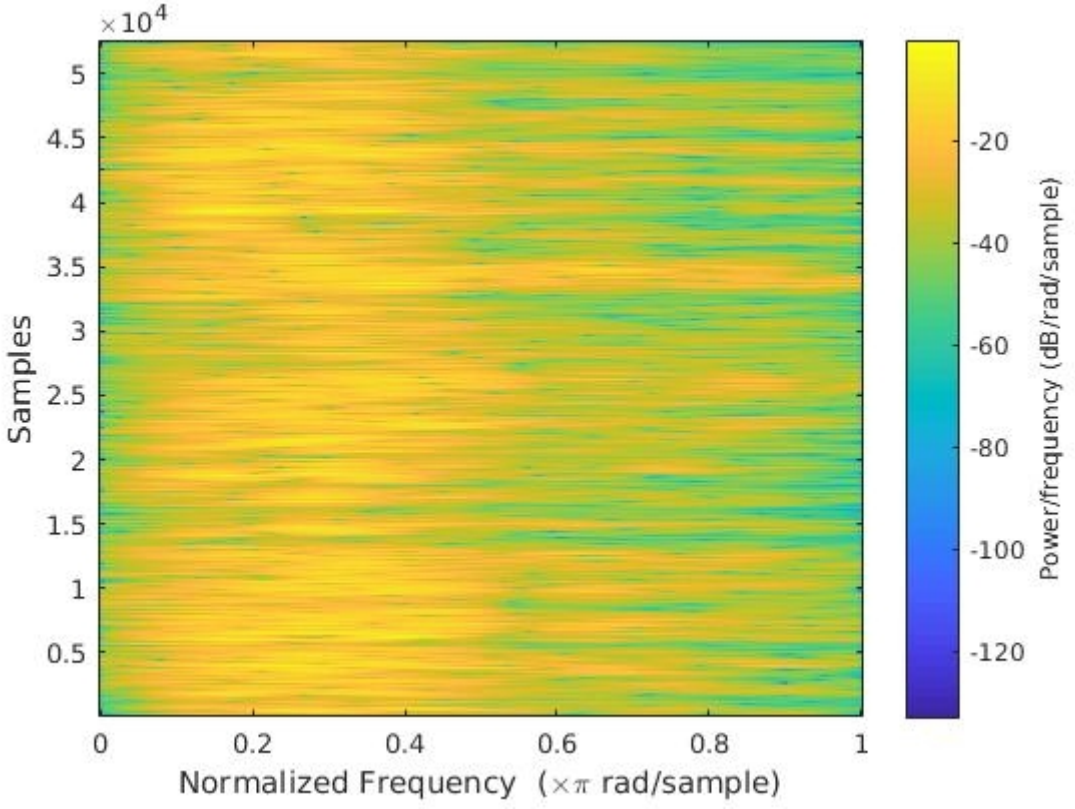


My spectrogram:

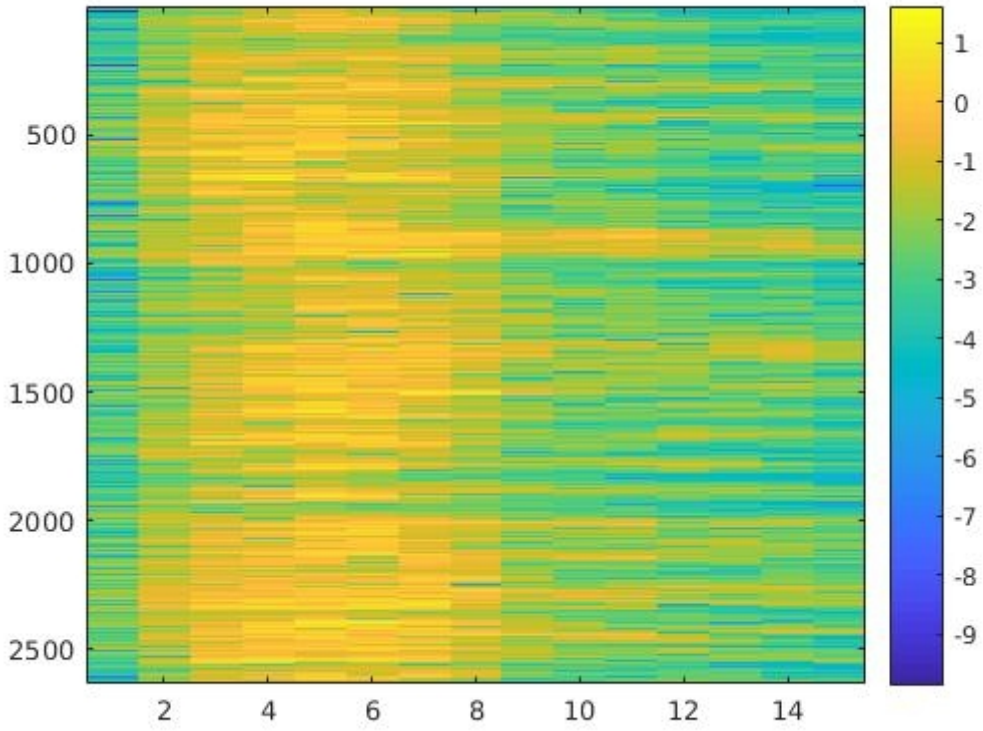


Laughter:

Inbuilt spectrogram:



My spectrogram:



Oberservations:

If the window increases in length then you have fewer windows spread across your audio and therefore a greater time interval between each - i.e. lower temporal resolution.

The coefficient (frequency resolution) quality, however, increases with window length. If you perform a fourier transform over a longer sample, you get a higher frequency resolution.

Hence, increase in the window length gives better frequency resolution vs decreasing time interval resolution and vice-versa.

This varies for each given overlap value. Hence, by keeping and overlap value constand and changing the window sizes, I came to the above conclusion.

4) Comparing between the 1a and 1b images of fft, we see that both of them look like the sinc function with the frequency bands slowly decreasing if we look at them with precision or using the zoom tool and measuring the coordinates. Also, the rectangular portion is merely translated and hence it doesn't causes change in the image to an extent. Both their fft plots look the same but since the rectangular portions aren't exactly the same and hence there is a slight difference in the curvatures of the central peak. Also, the amplitude slightly varies.

Comparing between the 2a and 2b images of fft, we know that the 2b image is a bit blurred or softened (and a cosine signal fused to it) as compared to the 2a image and hence, if we notice the high frequency spot near the
centre is smaller in the 2b as that in 2a. Also, there less number of high frequency stripes in the fft of 2b as that in 2a which also speaks of the same, which is significant from the image 2a and 2b. Looking at the plot of the fft of 2a and 2b, we see that 2b has an extra frequency as compared to 2a and 2b has lesser amplitudes than 2a.

5) I have stored all the ffts of the dialtones. Now, while identifying the telephone number, digit by digit. I take the dot product with the stored ffts of every number. And the one which gives me the maximum return value is the best match with the input. And hence, that digit which gave the maximum value is the answer digit. In the same way, we find the best match for the digits and return the best possible telephone number.

6) For this question, we use the brute force technique. Here, we use all the possible permutations of [1 2 3 4] and permute the the (sub)ffts of the input. The mirror ffts of the permutes are also adjusted in the same way.

For example, if the permutation is 2 1 4 3, then the conjugate part [5 6 7 8] is permuted as 6 5 8 7. And then, listening to all the permutations of the ffts and hence, the sounds, we decrpyt the message and find out the message from the best possible permutation.