EE3025 ASSIGNMENT- 1

SAI KARTHIK R - EE18BTECH11037

Download all python codes from

```
https://github.com/karthik−ramneti/ee3025/tree/
    main/assignment1c/codes
```

and latex-tikz codes from

```
https://github.com/karthik−ramneti/ee3025/tree/
    main/assignment1c
```

1 FFT Algorithm

CooleyTukey FFT algorithm divides a DFT of size N into two interleaved DFTs of size N/2 with each recursive stage producing an overall runtime of $\mathbf{O}(N\log(N))$.

Fourier transform of $(x = x_0, x_1, x_2, x_3, ..., x_n)$ is,

$$X_k = \sum_{i=0}^{N-1} x_i e^{\frac{-j2\pi}{N} ik} \tag{1.0.1}$$

We first compute the DFTs of the even-indexed inputs $(x_{2m} = x_0, x_2, \ldots, x_{N-2})$ and of the odd-indexed inputs $(x_{2m+1} = x_1, x_3, \ldots, x_{N-1})$, and then combines those two results to produce the DFT of the whole sequence.

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{N}2mk} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{N}(2m+1)k} \tag{1.0.2}$$

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{N}2mk} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{N}2mk} e^{\frac{-j2\pi}{N}k} \tag{1.0.3}$$

$$X_k = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{\frac{N}{2}}mk} + \left(e^{\frac{-j2\pi}{N}k}\right) \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{\frac{N}{2}}mk} \tag{1.0.4}$$

where,

$\sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{\frac{N}{2}}mk}$ is the fourier transform of $(x_{2m} = x_0, x_2, \ldots, x_{N-2})$ denoted by E_k.

$\sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{\frac{N}{2}}mk}$ is the fourier transform of $(x_{2m+1} = x_1, x_3, \ldots, x_{N-1})$ denoted by O_k.

$$\implies X_k = E_k + \left(e^{\frac{-j2\pi}{N}k}\right) O_k \tag{1.0.5}$$

Due to the periodocity of the complex exponential,

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{N}2m\left(k+\frac{N}{2}\right)} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{N}(2m+1)\left(k+\frac{N}{2}\right)} \tag{1.0.6}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{N}2m\left(k+\frac{N}{2}\right)} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{N}2m\left(k+\frac{N}{2}\right)} e^{\frac{-j2\pi}{N}\left(k+\frac{N}{2}\right)} \tag{1.0.7}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{N}2mk} e^{-j2\pi m} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{N}2mk} e^{-j2\pi m} e^{\frac{-j2\pi}{N}k} e^{-j\pi} \tag{1.0.8}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{N}2mk} + \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{N}2mk} e^{\frac{-j2\pi}{N}k} \times -1 \tag{1.0.9}$$

$$X_{k+\frac{N}{2}} = \sum_{m=0}^{\frac{N}{2}-1} x_{2m} e^{\frac{-j2\pi}{\frac{N}{2}}mk} - \left(e^{\frac{-j2\pi}{N}k}\right) \sum_{m=0}^{\frac{N}{2}-1} x_{2m+1} e^{\frac{-j2\pi}{\frac{N}{2}}mk} \tag{1.0.10}$$

$$\implies X_{k+\frac{N}{2}} = E_k - \left(e^{\frac{-j2\pi}{N}k}\right) O_k \tag{1.0.11}$$

$$\begin{bmatrix} X_0 \\ X_1 \\ X_2 \\ X_3 \\ . \\ . \\ . \\ X_{N/2} \end{bmatrix} = \begin{bmatrix} E_0 \\ E_1 \\ E_2 \\ E_3 \\ . \\ . \\ . \\ E_{N/2} \end{bmatrix} + \begin{bmatrix} W_N^0 & 0 & . & . & . & . \\ 0 & W_N^1 & 0 & 0 & . & . \\ 0 & 0 & W_N^2 & 0 & . & . \\ 0 & 0 & 0 & W_N^3 & . & . \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ 0 & 0 & . & . & . & W_N^{N/2} \end{bmatrix} \begin{bmatrix} O_0 \\ O_1 \\ O_2 \\ O_3 \\ . \\ . \\ . \\ O_{N/2} \end{bmatrix} \tag{1.0.12}$$

$$\begin{bmatrix} X_{N/2+1} \\ X_{N/2+2} \\ X_{N/2+3} \\ X_{N/2+4} \\ . \\ . \\ . \\ X_N \end{bmatrix} = \begin{bmatrix} E_0 \\ E_1 \\ E_2 \\ E_3 \\ . \\ . \\ . \\ E_{N/2} \end{bmatrix} - \begin{bmatrix} W_N^0 & 0 & . & . & . & . \\ 0 & W_N^1 & 0 & 0 & . & . \\ 0 & 0 & W_N^2 & 0 & . & . \\ 0 & 0 & 0 & W_N^3 & . & . \\ . \\ . \\ . \\ 0 & 0 & . & . & . & W_N^{N/2} \end{bmatrix} \begin{bmatrix} O_0 \\ O_1 \\ O_2 \\ O_3 \\ . \\ . \\ . \\ O_{N/2} \end{bmatrix} \tag{1.0.13}$$

This way fourier transform is computed by two sub fourier transforms, and these sub fourier transforms are calculated using the recursive calls.
This algorithm is only valid when the size of the input is a power of 2,so we make the size of the input to its nearest power of 2 by padding zeroes to it.

2 IFFT Algorithm

Similar to the FFT algorithm CooleyTukey IFFT algorithm divides a IDFT of size N into two interleaved IDFTs of size N/2 with each recursive stage producing an overall runtime of $\mathbf{O}(N\log(N))$.

Inverse fourier transform of
$(X = X_0, X_1, X_2, X_3, ..., X_n)$ is,

$$x_k = \frac{1}{N}\sum_{i=0}^{N-1} X_i e^{\frac{j2\pi}{N}ik} \tag{2.0.1}$$

We first compute the IDFTs of the even-indexed inputs $(X_{2m} = X_0, X_2, \ldots, X_{N-2})$ and of the odd-indexed inputs $(X_{2m+1} = X_1, X_3, \ldots, X_{N-1})$, and then combines those two results to produce the IDFT of the whole sequence.

$$x_k = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{N}2mk} + \sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{N}(2m+1)k}\right) \tag{2.0.2}$$

$$x_k = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{N}2mk} + \sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{N}2mk} e^{\frac{j2\pi}{N}k}\right) \tag{2.0.3}$$

$$x_k = \frac{1}{2}\left(\frac{1}{\frac{N}{2}}\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{\frac{N}{2}}mk} + \left(e^{\frac{j2\pi}{N}k}\right)\frac{1}{\frac{N}{2}}\sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{\frac{N}{2}}mk}\right) \tag{2.0.4}$$

where,
$\frac{1}{\frac{N}{2}}\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{\frac{N}{2}}mk}$ is the inverse fourier transform of $(X_{2m} = X_0, X_2, \ldots, X_{N-2})$ denoted by e_k.
$\frac{1}{\frac{N}{2}}\sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{\frac{N}{2}}mk}$ is the inverse fourier transform of $(X_{2m+1} = X_1, X_3, \ldots, X_{N-1})$ denoted by o_k.

$$\implies x_k = \frac{1}{2}\left(e_k + \left(e^{\frac{j2\pi}{N}k}\right)o_k\right) \tag{2.0.5}$$

Due to the periodocity of the complex exponential,

$$x_{k+\frac{N}{2}} = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{N}2m\left(k+\frac{N}{2}\right)} + \sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{N}(2m+1)\left(k+\frac{N}{2}\right)}\right) \tag{2.0.6}$$

$$x_{k+\frac{N}{2}} = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{N}2m\left(k+\frac{N}{2}\right)} + \sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{N}2m\left(k+\frac{N}{2}\right)} e^{\frac{j2\pi}{N}\left(k+\frac{N}{2}\right)}\right) \tag{2.0.7}$$

$$x_{k+\frac{N}{2}} = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{N}2mk} e^{j2\pi m} + \sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{N}2mk} e^{j2\pi m} e^{\frac{j2\pi}{N}k} e^{j\pi}\right) \tag{2.0.8}$$

$$x_{k+\frac{N}{2}} = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{N}2mk} + \sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{N}2mk} e^{\frac{j2\pi}{N}k} \times -1\right) \tag{2.0.9}$$

$$x_{k+\frac{N}{2}} = \frac{1}{N}\left(\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{\frac{N}{2}}mk} - \left(e^{\frac{j2\pi}{N}k}\right)\sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{\frac{N}{2}}mk}\right) \tag{2.0.10}$$

$$x_{k+\frac{N}{2}} = \frac{1}{2}\left(\frac{1}{\frac{N}{2}}\sum_{m=0}^{\frac{N}{2}-1} X_{2m} e^{\frac{j2\pi}{\frac{N}{2}}mk} - \left(e^{\frac{j2\pi}{N}k}\right)\frac{1}{\frac{N}{2}}\sum_{m=0}^{\frac{N}{2}-1} X_{2m+1} e^{\frac{j2\pi}{\frac{N}{2}}mk}\right) \tag{2.0.11}$$

$$\implies x_{k+\frac{N}{2}} = e_k - \left(e^{\frac{j2\pi}{N}k}\right)o_k \tag{2.0.12}$$

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \\ x_3 \\ . \\ . \\ . \\ x_{N/2} \end{bmatrix} = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ . \\ . \\ . \\ e_{N/2} \end{bmatrix} + \begin{bmatrix} W_N^{-0} & 0 & . & . & . & . \\ 0 & W_N^{-1} & 0 & 0 & . & . \\ 0 & 0 & W_N^{-2} & 0 & . & . \\ 0 & 0 & 0 & W_N^{-3} & . & . \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ 0 & 0 & . & . & . & W_N^{-N/2} \end{bmatrix} \begin{bmatrix} o_0 \\ o_1 \\ o_2 \\ o_3 \\ . \\ . \\ . \\ o_{N/2} \end{bmatrix} \tag{2.0.13}$$

$$\begin{bmatrix} x_{N/2+1} \\ x_{N/2+2} \\ x_{N/2+3} \\ x_{N/2+4} \\ . \\ . \\ . \\ x_N \end{bmatrix} = \begin{bmatrix} e_0 \\ e_1 \\ e_2 \\ e_3 \\ . \\ . \\ . \\ e_{N/2} \end{bmatrix} - \begin{bmatrix} W_N^{-0} & 0 & . & . & . & . \\ 0 & W_N^{-1} & 0 & 0 & . & . \\ 0 & 0 & W_N^{-2} & 0 & . & . \\ 0 & 0 & 0 & W_N^{-3} & . & . \\ . & & & & & \\ . & & & & & \\ . & & & & & \\ 0 & 0 & . & . & . & W_N^{-N/2} \end{bmatrix} \begin{bmatrix} o_0 \\ o_1 \\ o_2 \\ o_3 \\ . \\ . \\ . \\ o_{N/2} \end{bmatrix} \tag{2.0.14}$$

This way inverse fourier transform is computed by two sub inverse fourier transforms, and these sub inverse fourier transforms are calculated using the recursive calls.

3 Problem

The command

```
output_signal = signal.lfilter(b,a,
    output_signal)
```

in Problem 2.3 is executed through following difference equation

$$\sum_{m=0}^{M} a(m)\, y(n-m) = \sum_{k=0}^{N} b(k)\, x(n-k) \tag{3.0.1}$$

where input signal is $x(n)$ and output signal is $y(n)$ with intial values all 0. Replace **signal.filtfilt** with your own routine and verify

4 Solution

Let $X(z)$ and $Y(z)$ be the respective z-transforms of $x(n)$ and $y(n)$ respectively. Using the properties of z-transform

$$\mathcal{Z}\{x(n-k)\} = z^{-k}X(z) \tag{4.0.1}$$

$$\mathcal{Z}\{y(n-m)\} = z^{-m}Y(z) \tag{4.0.2}$$

Applying z-transform to the both sides of the difference equation

$$Y(z)\left(\sum_{m=0}^{M} a(m)\, z^{-m}\right) = X(z)\left(\sum_{k=0}^{N} b(k)\, z^{-k}\right) \tag{4.0.3}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b(k)\, z^{-k}}{\sum_{m=0}^{M} a(m)\, z^{-m}} \tag{4.0.4}$$

$H(K)$ is evaluated from (4.0.4) using the coefficients b,a.

$X(z)$ is evaluated by the above mentioned fft algorithm.

$Y(z)$ is evaluated by multiplying $X(z)$ and $H(z)$.

$$Y(K) = H(K)\, X(K) \tag{4.0.5}$$

$y(n)$ is evaluated by the above mentioned ifft algorithm.

5 Implementation in C

At first x.dat file is created from the given sound file. Next H.dat file is generated which is the DFT of the filters impulse response, that is computed using the filter coefficients b,a. The following code is used to generate these files

```
codes/generate.py
```

Then we read x.dat file and obtain x and compute the fft of x. Then by mutiplying it with H we get

Y. Y is stored in Y.dat file and the time domain signal is stored in y.dat file. This is done by folling c program.

```
codes/fft.c
```

Compile the program by

```
gcc fft.c −o fft −lm
```

Run the program by

```
./fft
```

Finally the stored files are used for generating the new sound file, by the following python code.

```
codes/ee18btech11037.py
```

The sound file obtained using c routine

```
codes/Sound_With_ReducedNoise_ownroutine.
    wav
```

The sound file obtained using inbuilt command

```
codes/Sound_With_ReducedNoise.wav
```

6 Verification

Both the ouput signals obtained using builtin signal.filtfilt command and own routine method sounds the same.
Plotting the time domain output signal evaluated from both own routine filter and signal.filtfilt command
Plotting the frequency domain response evaluated from both own routine and signal.filtfilt

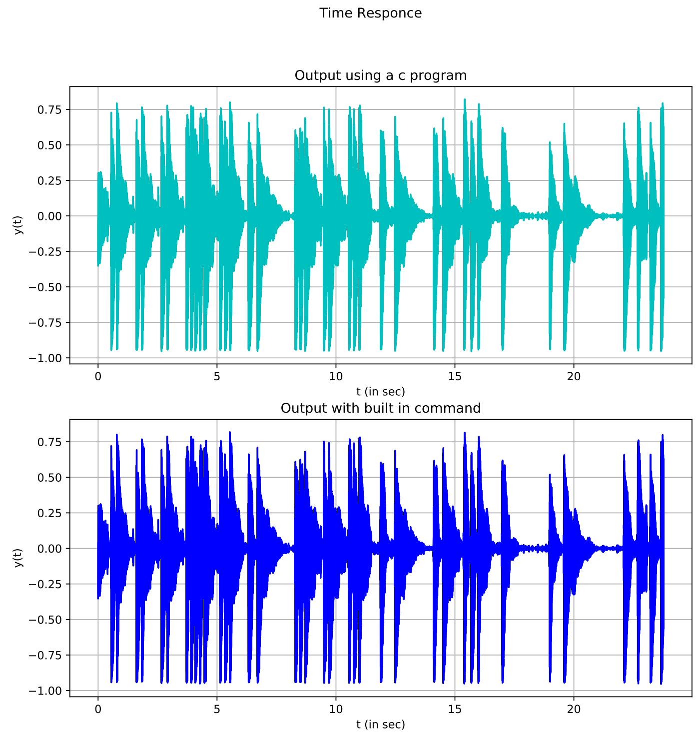

Fig. 0: Time domain response

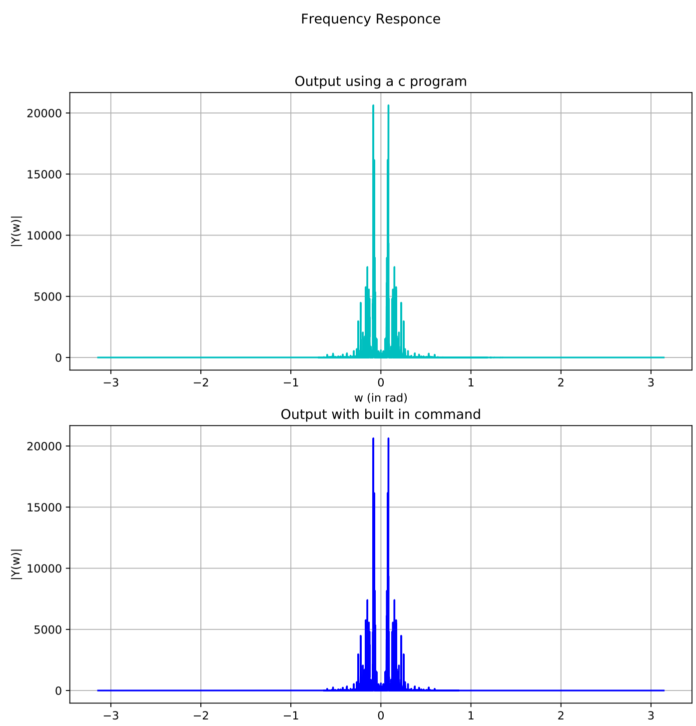

Fig. 0: Frequency domain response