FFT Implementation

Neil Dhami - EE18BTECH11031

Download all codes from

```
https://github.com/neildhami18/IITH_Academics/EE3025/Assignment2/codes
```

and latex-tikz codes from

```
https://github.com/neildhami18/IITH_Academics/EE3025/Assignment1
```

1 Problem

The command

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

in Problem 2.3 is executed through following difference equation

$$\sum_{m=0}^{M} a(m)\, y(n-m) = \sum_{k=0}^{N} b(k)\, x(n-k) \tag{1.0.1}$$

where input signal is $x(n)$ and output signal is $y(n)$ with initial values all 0. Replace **numpy.fft** used in previous assignment with **your own FFT routine coded in C** and verify.

2 Method : The Cooley-Tukey Implementation

The N-point Discrete Fourier Transform (DFT) for a given sequence $x[n]$ is defined by the formula:

$$X(k) = \sum_{n=0}^{N-1} x[n] W_N^{kn} \tag{2.0.1}$$

$$\text{where,} W_N = e^{\frac{-2\pi i}{N}} \tag{2.0.2}$$

$$\tag{2.0.3}$$

The radix-2 DIT approach to Cooley-Tukey algorithm rearranges this DFT into two parts: a sum over the even-numbered indices $n = 2r$ and a sum over the odd-numbered indices $n = 2r + 1$:

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} x[2r] W_N^{(2r)k} + \sum_{r=0}^{\frac{N}{2}-1} x[2r+1] W_N^{(2r+1)k} \tag{2.0.4}$$

$$\text{Where, } x[2r] = e[r],\ x[2r+1] = o[r] \tag{2.0.5}$$

$$\tag{2.0.6}$$

The equation.2.0.4 can be rewritten as the following:

$$X(k) = \sum_{r=0}^{\frac{N}{2}-1} e[r] W_{N/2}^{kr} + W_N^k \sum_{r=0}^{\frac{N}{2}-1} o[r] W_{N/2}^{kr} \tag{2.0.7}$$

using the exponential property of complex numbers:

$$W_{\frac{N}{2}} = W_N^2 \tag{2.0.8}$$

Assuming N is even, equation (2.0.7) illustrates a summation of two N/2 point DFTs of the subsequences corresponding to even and odd positions respectively in the sequence $x[n]$. Therefore, the equation ∀ k in [0,N/2) can be written as:

$$X(k) = E(k) + W_N^k O(k) \tag{2.0.9}$$

Where E(k) is the dft of the even indices($e[n]$) and O(k) is the dft of the odd indices($o[n]$) of $x[n]$.
On exploiting the property of periodicity, ∀ k+N/2 in [N/2,N) the equation can be written as:

$$X(k + N/2) = E(k + N/2) + W_N^{k+N/2} O(k + N/2) \tag{2.0.10}$$

which, on substituting in (2.0.7), simplifies to:

$$X(k + N/2) = E(k) - W_N^k O(k) \tag{2.0.11}$$

From the equations.(2.0.9) and (2.0.11),
for k in [0,N/2)..,

$$F_N(x[n]) = F_{N/2}(e[n]) + F_{N/2} D_{N/2}(o[n]) \tag{2.0.12}$$

for k in [N/2,N)..,

$$F_N(x[n]) = F_{N/2}(e[n]) - F_{N/2} D_{N/2}(o[n]) \tag{2.0.13}$$

where D_N is the diagonal matrix with diagonal values $[1, W_N^1, W_N^2, W_N^3..., W_N^{N-1}]$.
Combining the above two equations,

$$F_N x[n] = \begin{bmatrix} I_{N/2} & D_{N/2} \\ I_{N/2} & -D_{N/2} \end{bmatrix} \begin{bmatrix} F_{N/2} & 0 \\ 0 & F_{N/2} \end{bmatrix} \begin{bmatrix} e[n] \\ o[n] \end{bmatrix} \tag{2.0.14}$$

Thus, we can compute F_N from $F_{N/2}$, $F_{N/2}$ from $F_{N/4}$. This is the recursive approach with N=1 being the base case. When N = 1, FFT(x) = x.

3 Solution

The approach is similar to what we did in Assignment-1.
From the time shifting property of Z transfrom,

$$\mathcal{Z}\{x(n-k)\} = z^{-k}X(z) \tag{3.0.1}$$

$$\mathcal{Z}\{y(n-m)\} = z^{-m}Y(z) \tag{3.0.2}$$

where $X(z)$ and $Y(z)$ are the z-transforms of $x(n)$ and $y(n)$ respectively.
The equation obtained in Z domain:

$$Y(z) \sum_{m=0}^{M} a(m) z^{-m} = X(z) \sum_{k=0}^{N} b(k) z^{-k} \tag{3.0.3}$$

$$H(z) = \frac{Y(z)}{X(z)} = \frac{\sum_{k=0}^{N} b(k) z^{-k}}{\sum_{m=0}^{M} a(m) z^{-m}} \tag{3.0.4}$$

From the coefficients b,a and from (3.0.4) we evaluate $H(k)$ as:

$$H(k) = H\left(z = e^{-\jmath 2\pi k/N}\right). \tag{3.0.5}$$

The following python code stores the sound signal $x(n)$ as well as transfer function $H(jw)$.

```
codes/ee18btech11031−fft−inputdata.py
```

Now, we perform the following steps to obtain the output:

$$X(k) = fft(x(n)) \tag{3.0.6}$$

$$Y(k) = H(k)X(k) \tag{3.0.7}$$

$$y(n) = ifft(Y(k)) \tag{3.0.8}$$

The following C code performs these operations and saves the data..

```
codes/ee18btech11031−fft−main.c
```

4 Verification

The following python code collects data from the C-program and writes the output soundfile.

```
codes/ee18btech11031−fft−outputdata.py
```

Below is the audio file for the output y(n)

```
codes/Sound_With_ReducedNoise_cfft.wav
```

Plotting and comparing the time domain output signal $y(n)$ obtained from inbuilt functions in python and the own routine in C..

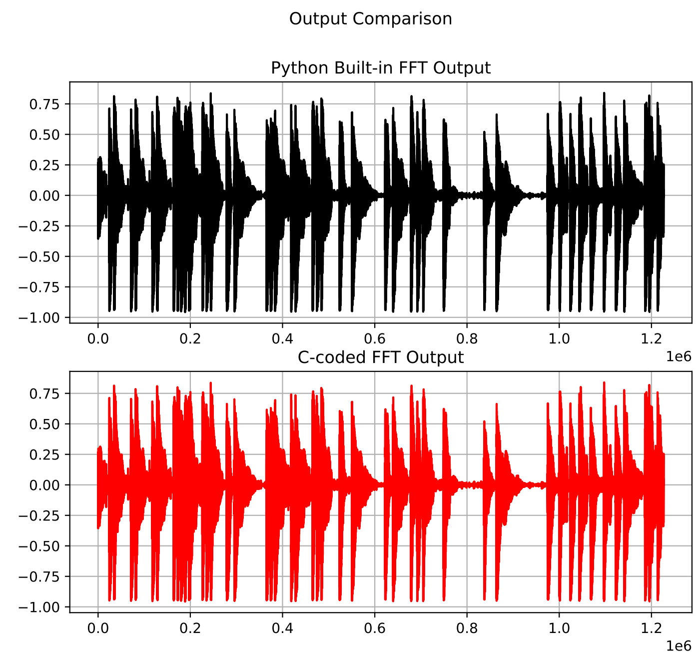

Fig. 0: Time domain response