Signals & Systems Project

(Signals & Systems Course - Fall 2022)

Contributor

Matin Monshizadeh (9932122)

Part A:

Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h1[n]$:

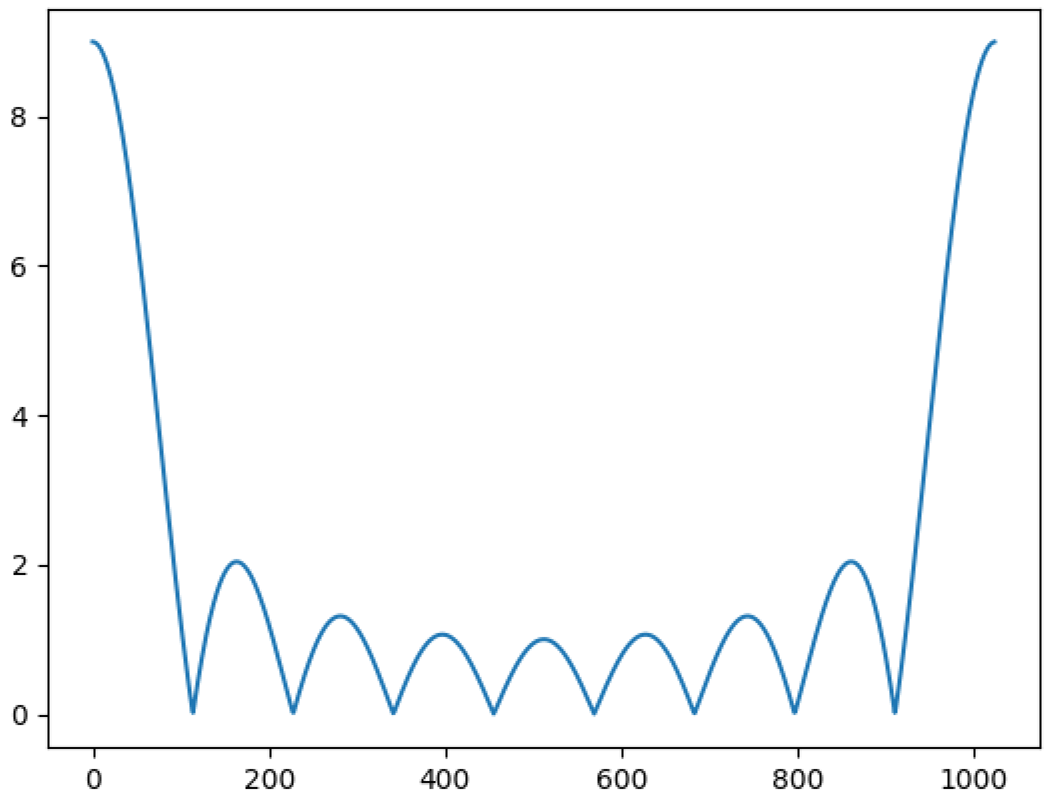
In this part, we use the *Matplotlib* library to show results as charts and use the *Numpy* library for numerical calculus.

```python
def h1Function(n, N1):
    if n >= 0 and n <= 2 * N1:
        return 1
    else:
        return 0
```

- At first, we declare h1Function, a function to return the value of $h1[n]$.
- The value of h1Function is one if n between 0 and $2 * N1$; otherwise, the value is zero.
- Question information said that $N1$ is equal to four.

```python
def DFT(h1Function, K):
    result = []
    for i in range(K):
        result += [0]
    for k in range(K):
        for n in range(-1000, 1000):
            result[k] += h1Function(n, 4) * np.exp(2j * np.pi * k * n / K)
    return result
```

- Also, we declare a DFT function that returns the discrete Fourier transform of $h1[n]$.
- $H1(k) = \sum_{n=-1000}^{1000} h1[n]e^{\frac{j2\pi kn}{K}}$
- As we know that we can't define infinite, we specify the scope of n between -1000 and 1000.
- Question information said that K is equal to 1024.



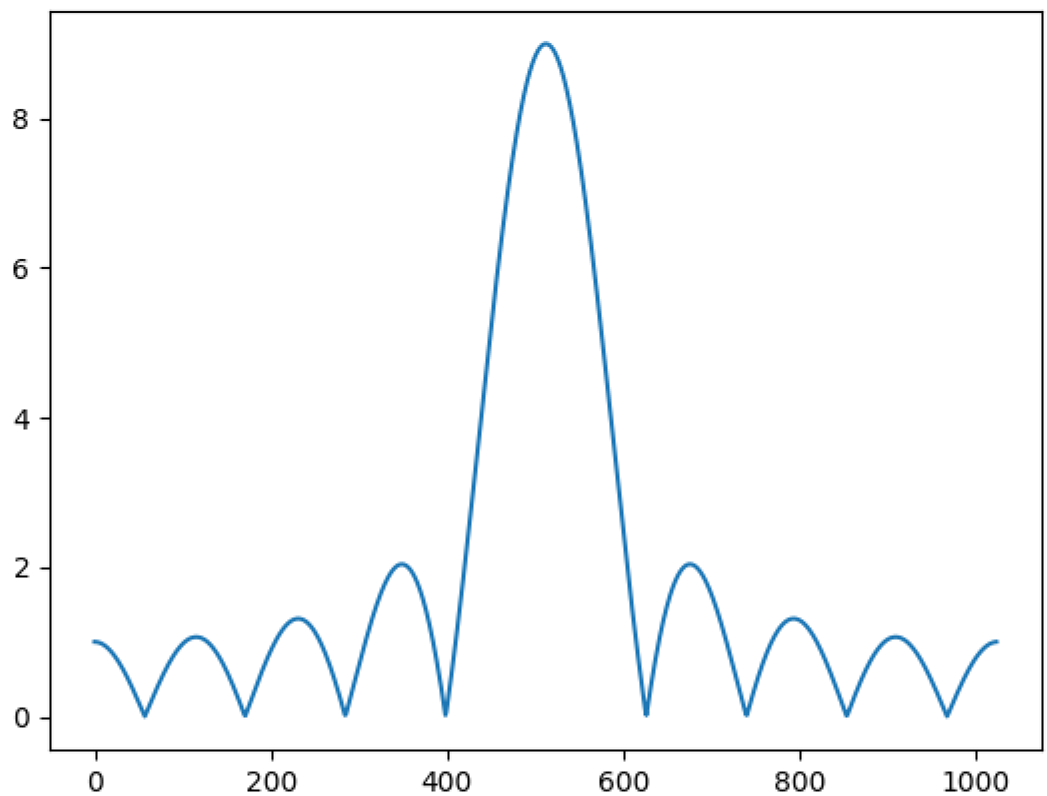
0
2
4
6
8
0
200
400
600
800
1000

Part B:

Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h2[n]$:

```
def h2Function(n, N1):
    if n >= 0 and n <= 2 * N1:
        return 1 * np.exp(1j * np.pi * n)
    return 0
```

- In this part, we do the same as the previous part, but we have h2Function that returns the value $1 * e^{j\pi n}$ if n between 0 and $2 * N1$; otherwise, the value is zero.
- The DFT function is the same as in the previous part.
- $H2(k) \;=\; \sum_{n=-1000}^{1000} h1[n]e^{j\pi n}e^{\frac{j2\pi kn}{K}}$

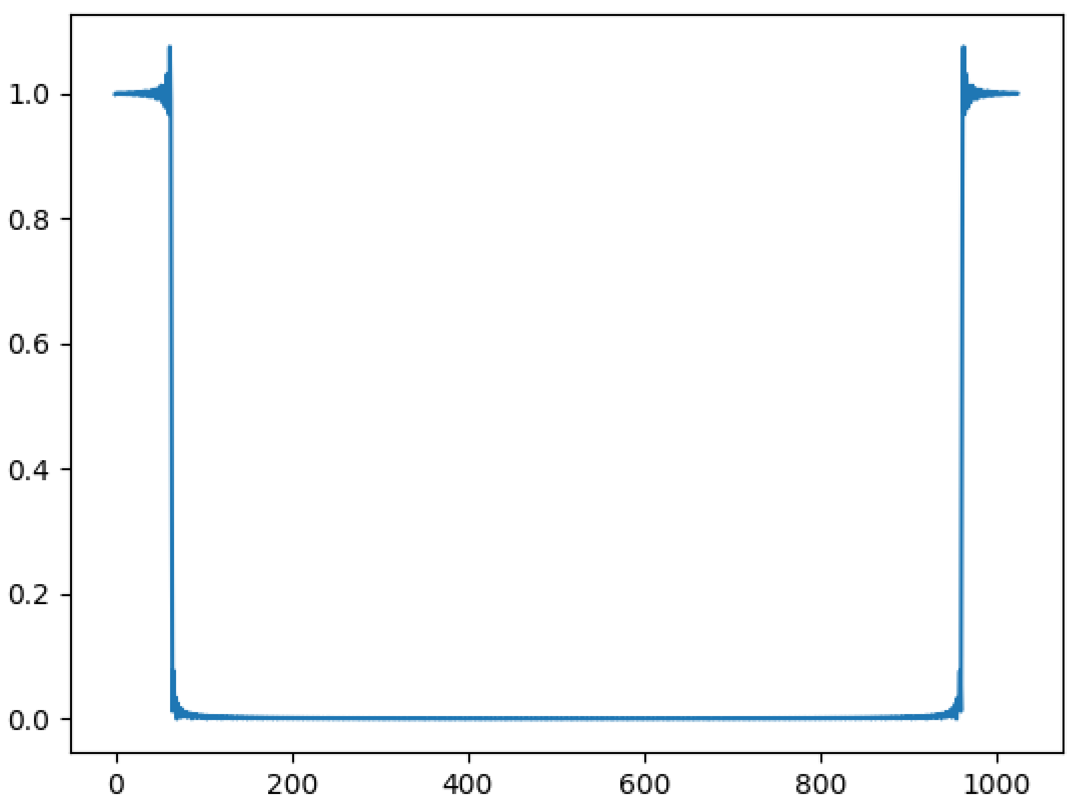


Part C:

Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h3[n]$:

```python
def h3Function(n, N2, omega):
    if n == 300:
        return 0.125
    if n >= 0 and n <= 2 * N2:
        return (np.sin(omega * (n - N2))) / (np.pi * (n - N2))
    return 0
```

- We do as in previous parts and declare h3Function.
- When n equals 300, our equation is ambiguous; then, the limit of the equation is 0.125.
- The DFT function is the same as in the previous part.
- $H3(k) \ = \sum_{n=-1000}^{1000} \frac{sin\ \omega(n-N2)}{\pi(n-N2)} e^{\frac{j2\pi kn}{K}}$



Part D:

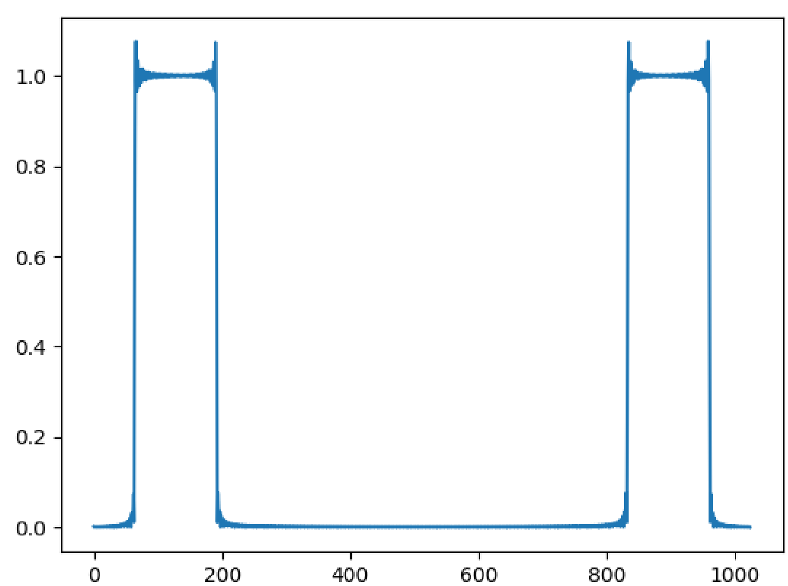
Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h4[n]$:

```python
def hFunction(n):
    N = 300
    omega = np.pi / 8
    # When n equals 300, our equation is ambiguous; then, the limit of the equation is 0.125
    if n == 300:
        return 0.125
    if n >= 0 and n <= 2 * N:
        return (np.sin(omega * (n - N))) / (np.pi * (n - N))
    return 0
```

- We do as in previous parts and declare hFunction.
- After that, we declare each function separately.
- The DFT function is the same as in the previous part.

```python
def h4Function(n):
    return 2 * np.cos((np.pi / 4) * n) * hFunction(n)
```

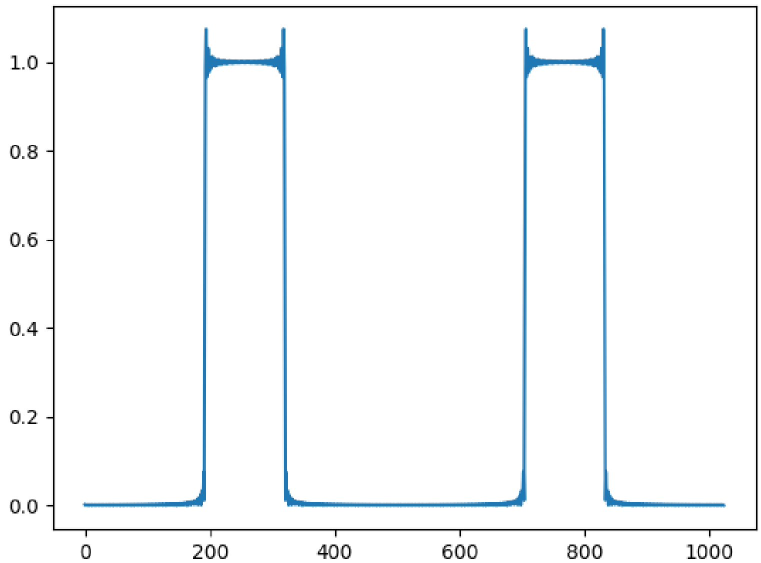
- $H4(k) \;=\; \sum_{n=-1000}^{1000} 2cos(\frac{\pi}{4}n)\frac{sin\,\omega(n-N2)}{\pi(n-N2)}e^{\frac{j2\pi kn}{K}}$



Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h5[n]$:

```python
def h5Function(n):
    return 2 * np.cos((np.pi / 2) * n) * hFunction(n)
```

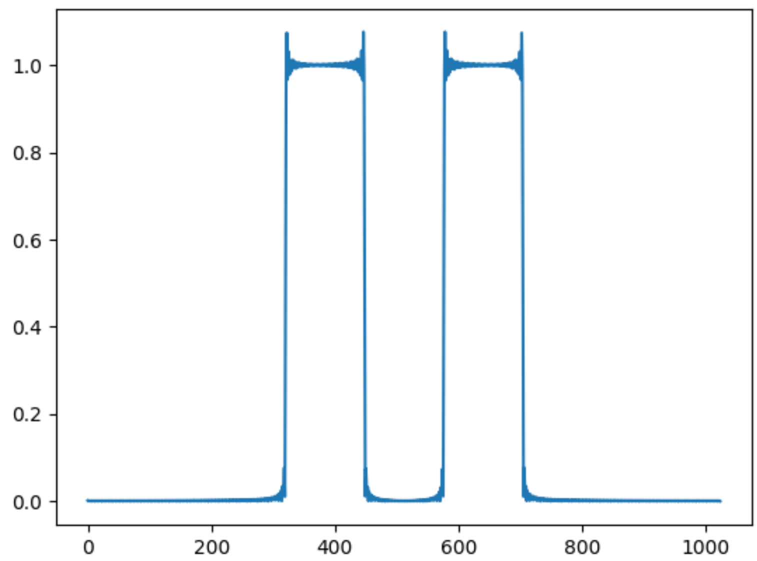
- $H5(k) = \sum_{n=-1000}^{1000} 2cos(\frac{\pi}{2}n)\frac{sin\,\omega(n-N2)}{\pi(n-N2)}e^{\frac{j2\pi kn}{K}}$



Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h6[n]$:

```python
def h6Function(n):
    return 2 * np.cos((3 * np.pi / 4) * n) * hFunction(n)
```

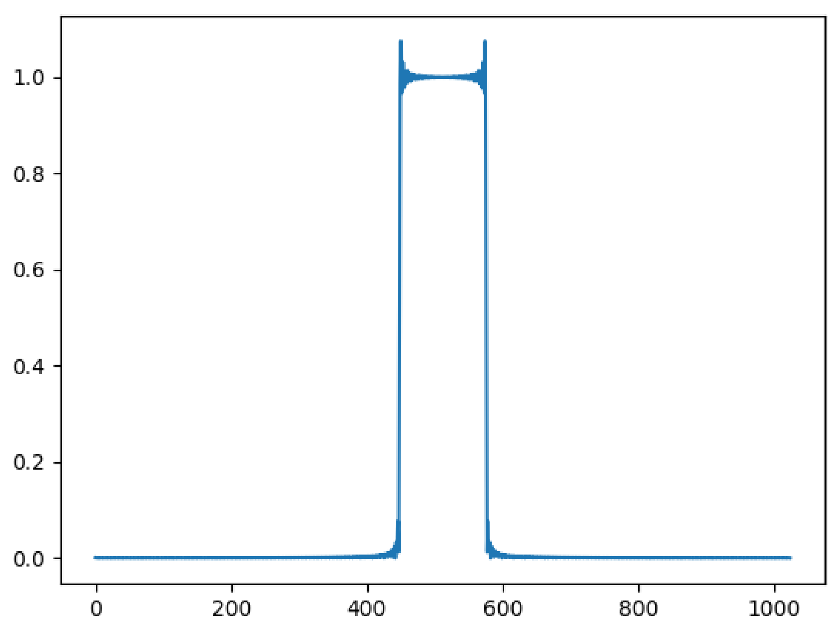
- $H6(k) = \sum_{n=-1000}^{1000} 2cos(\frac{3\pi}{4}n)\frac{sin\,\omega(n-N2)}{\pi(n-N2)}e^{\frac{j2\pi kn}{K}}$



Finding the <u>absolute value</u> of DFT(discrete fourie transform) of $h7[n]$:

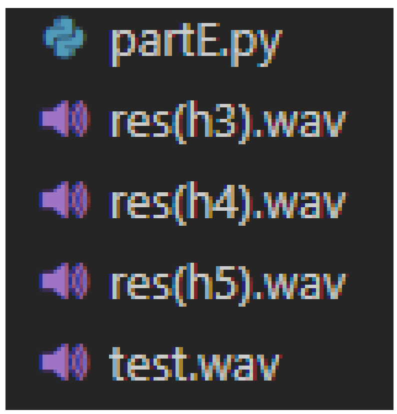
```
def h7Function(n):
    return np.cos((np.pi / 2) * n) * hFunction(n)
```

- $H7(k) \;=\; \sum_{n=-1000}^{1000} 2cos(\pi n)\frac{sin\,\omega(n-N2)}{\pi(n-N2)}e^{\frac{j2\pi kn}{K}}$



Part E:

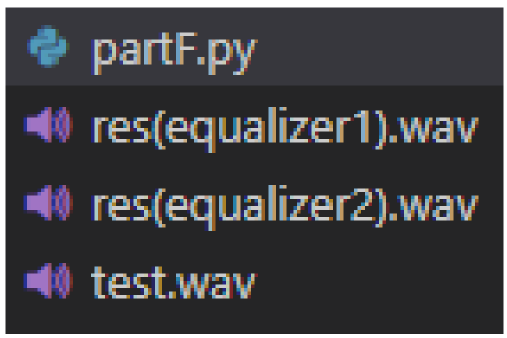
- First, we consider an audio file, then I change its extension to WAV and assume that its mode is mono-channel.
- After that we use the Scipy library to read audio files.
- Then create the arrays for the responses.
- Calculate the responses.
- Then convolve the data with the response.
- At the end write files.



- Our outputs.
- The change we observed between the outputs is that the higher the hit goes, the lower the sound becomes; as a result, the frequency increases.

Part F:

- We define all functions again.
- Then define equalizer 1 and 2 functions.
- we consider an audio file, then I change its extension to WAV and assume that its mode is mono-channel
- After that we use the Scipy library to read audio files.
- At the end write files.



- Our outputs.
- Due to the higher factor that hit has a lower frequency, equalizer 1 has a lower sound.