In [ ]:
```python
import numpy as np
import matplotlib.pyplot as plt
```

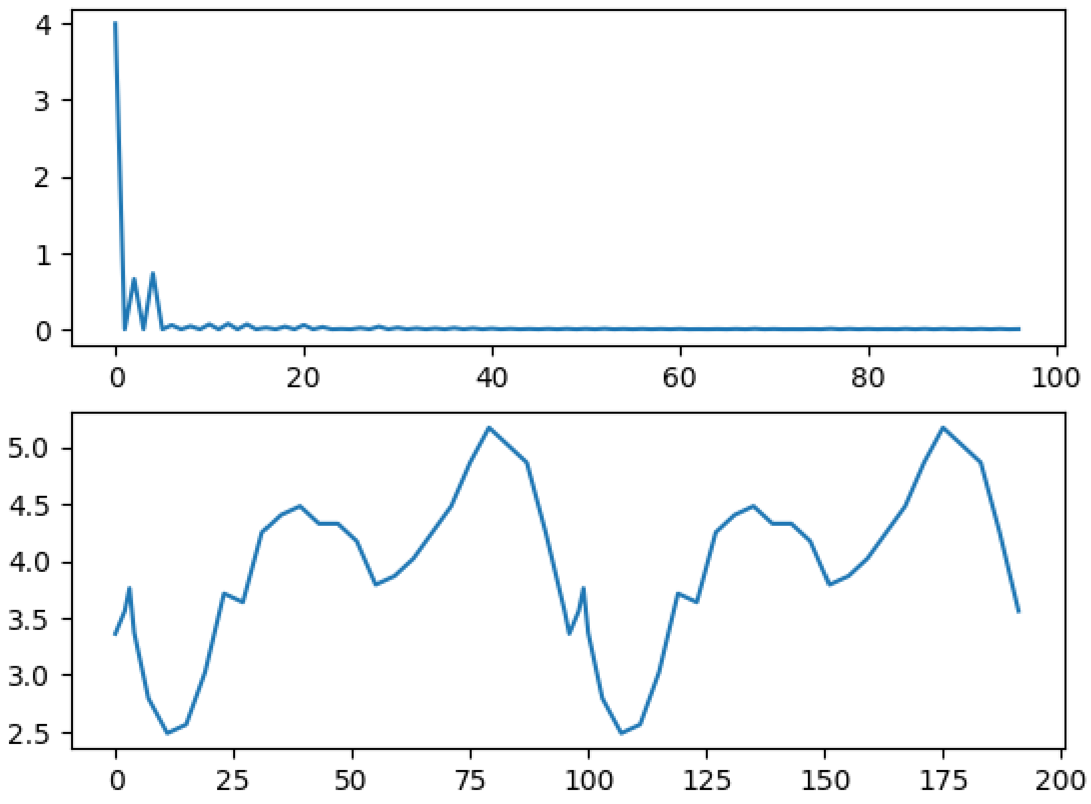
In [ ]:
```python
with open('freq_gas.npy', 'rb') as f:
    freq = np.load(f)

with open('time_gas.npy', 'rb') as f:
    time = np.load(f)
```

Data from gas_transmission.py

We plot the frequency and time from the given file.

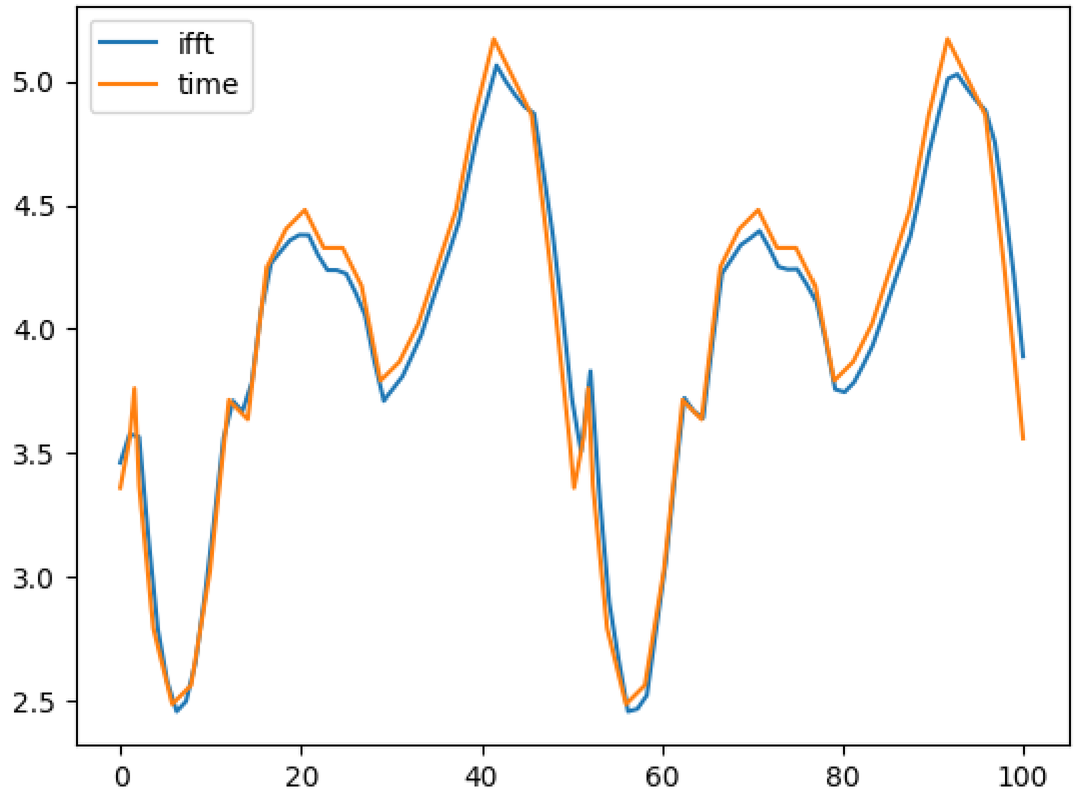
In [ ]:
```python
axs = plt.subplot(2, 1, 1)
axs.plot(np.abs(freq))
axs = plt.subplot(2, 1, 2)
axs.plot(time)
plt.show()
```



We can see that many frequencies are close to zero.

To ensure the data is extracted correctly from the file we will take the inverse fourier transform of the frequencies and compare it with the time series plot.

In [ ]:
```python
x = np.linspace(0, 100, len(freq))
x1 = np.linspace(0, 100, len(time))
plt.plot(x, abs(np.fft.ifft(freq))*95, label='ifft')
plt.plot(x1, time, label='time')
plt.legend()
plt.show()
```



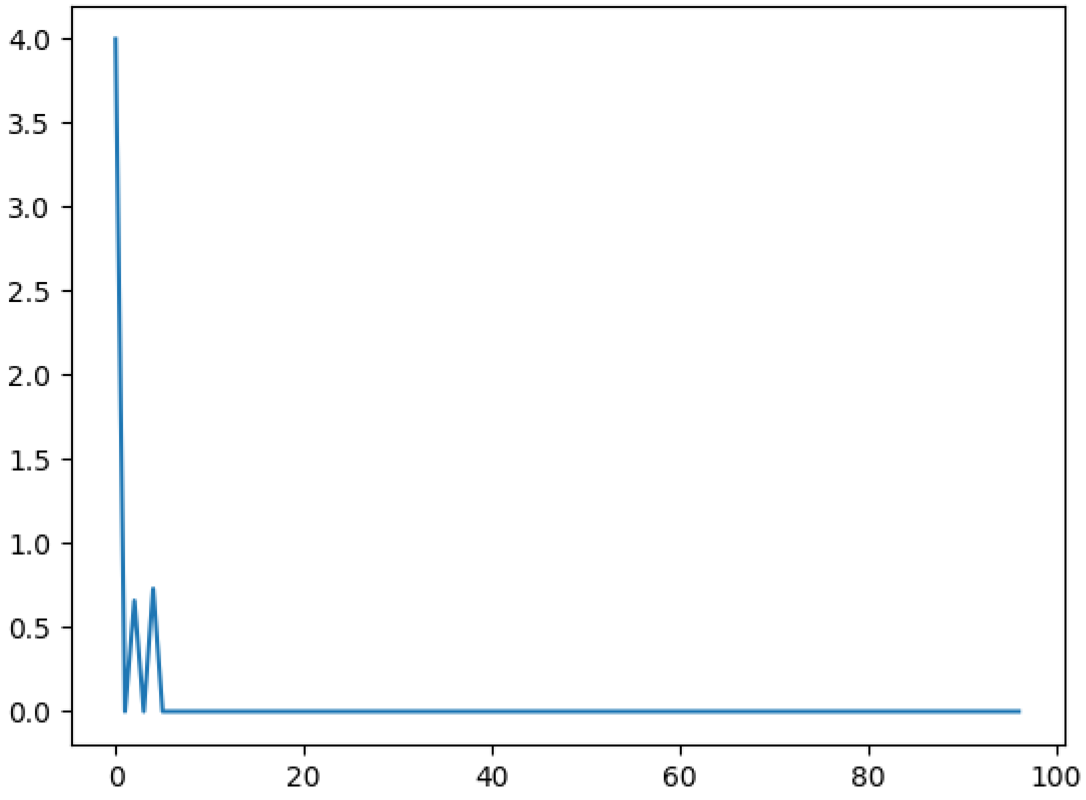
The inverse transformed frequencies are close to the time series data, but not exact. This might be due to numerical roundoff but it also seems like there is a buildin dimension reduction in their FFT implementation.

Frequency processing

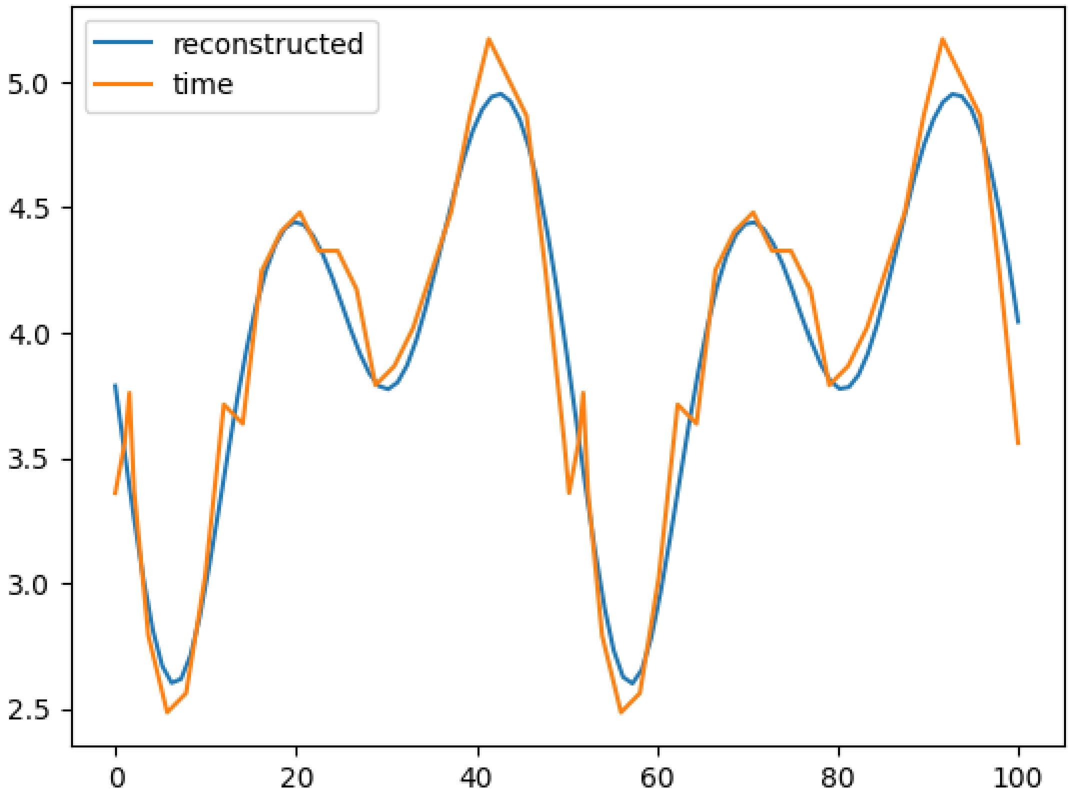
There are only about 3 frequencies that are fairly large, hence we will try to let the rest be equal to 0. It was suggested to only keep the 90% biggest frequencies, but this would only leave one value different from 0 and this is not what is wanted.

```
In [ ]: threshold = np.max(np.abs(freq))*0.9
        freq[np.abs(freq) < 0.5] = 0

        plt.plot(np.abs(freq))
        plt.show()
```



In [ ]:
```
x = np.linspace(0, 100, len(freq))
x1 = np.linspace(0, 100, len(time))
plt.plot(x, abs(np.fft.ifft(freq))*95, label='reconstructed')
plt.plot(x1, time, label='time')
plt.legend()
plt.show()
```



In [ ]: