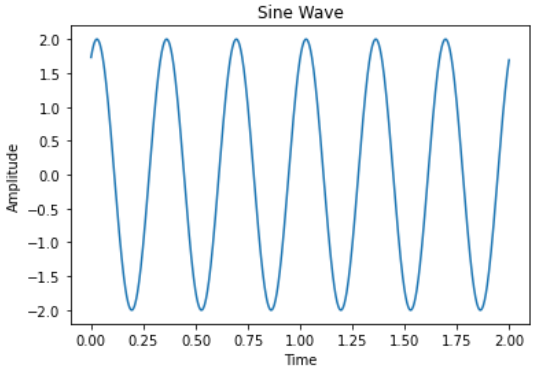
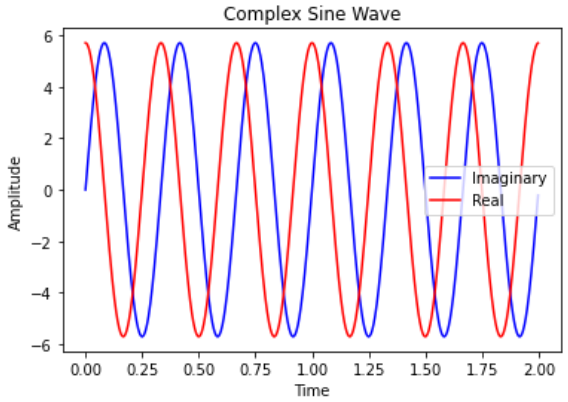
Question 1: Intro to Fourier Transform

Sine/Complex Sine Waves:

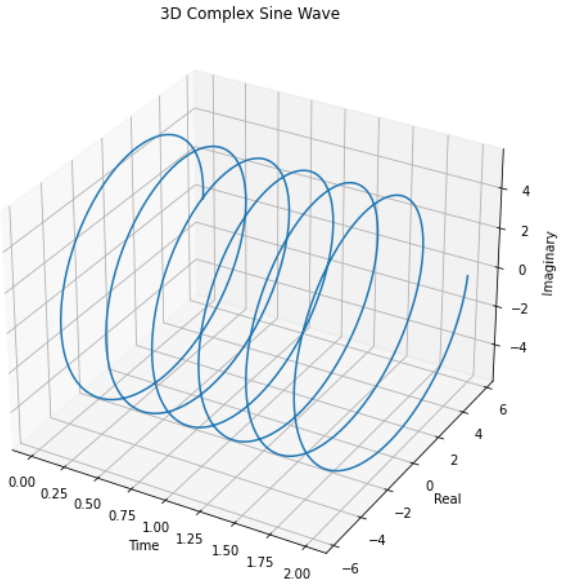
1-1A:



1-1B:



1-1C:



Dot/Complex-Dot-Product:

1-2A:

We can interpret dot product as sum of covariance of the vectors in a statically manner. This is a measure of how strongly two vectors share their pattern of values.

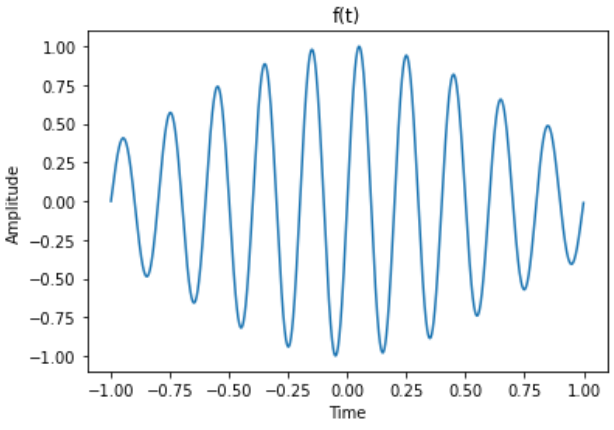
If the vectors are identical every value is squared, if they differ the result will be smaller than the square of the larger value. This means that the resulting value is useful for measuring how similar two vectors are and finally if the two vectors are orthogonal they share no common variation and the sum of the products cancel out resulting in zero.

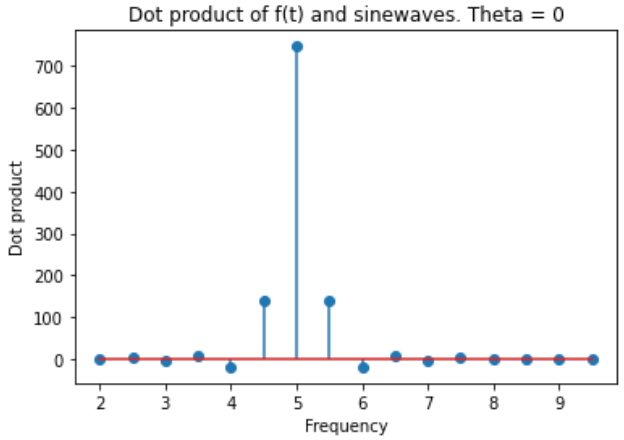
1-2B:

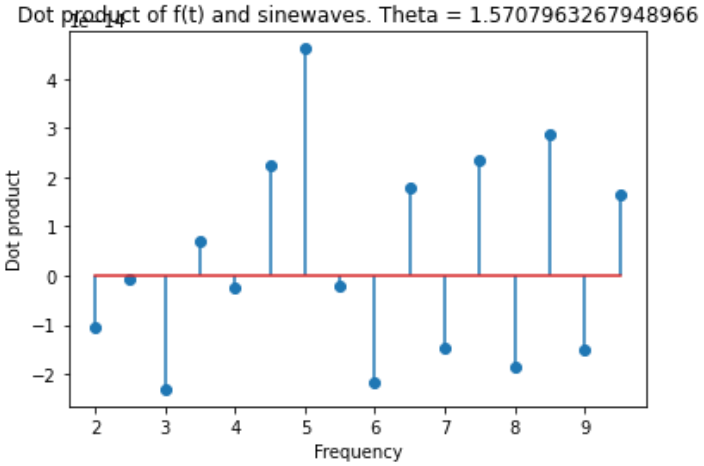
By changing frequency, the wave shifts thus making the waves different and eventually decreasing the dot product result since it's changing.

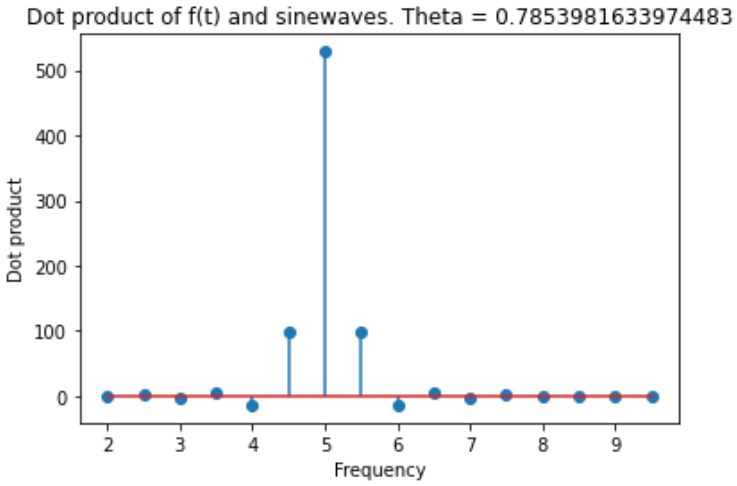
Again, by increasing the phase, dot product decreases. This is obvious since the shift in the sine wave will decrease the dot product because the waves become less identical.

1-2C:





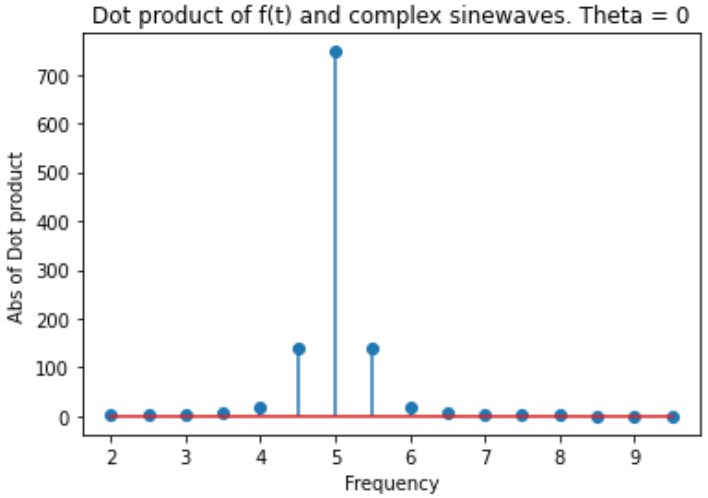


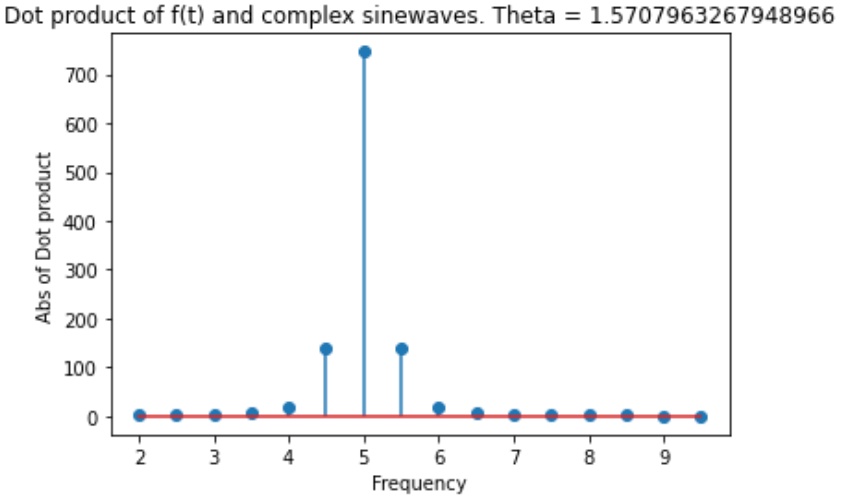


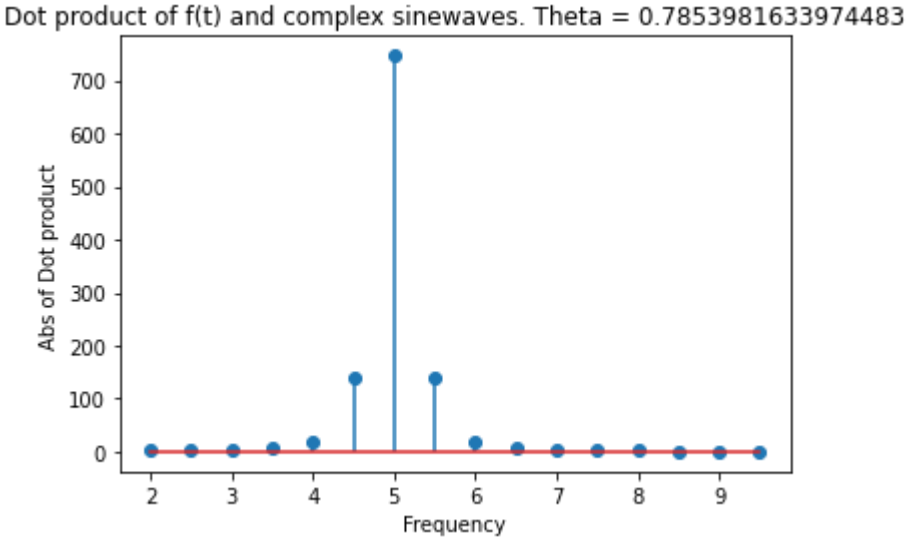
The maximum product is for the wave with frequency 5 which is the identical frequency to our initial function. As expected, by increasing phase, the dot product decreases. For $\frac{\pi}{4}$ shift, the maximum is still at frequency 5 but the value of the maximum product is lower than phase 0. For $\frac{\pi}{2}$shift, the maximum is still at frequency 5 but the value range for dot products are different. The maximum value is 4 in this phase. The current dot

product is problematic because it can change dramatically if we shift our signal, which is not that good.

1-2D:







As you can see, plots are identical to each other. This is because shifting in complex sin doesn't have any effects in abs of result($\left|e^{j\theta}\right| = 1$) so abs won't change with shifting.

Question 2: Fast Fourier Transform(fft)/Discrete Fourier Transform

Notes:

Sampling rate is 1000.

Nyquist Frequency: Nyquist frequency is a characteristic of a sampler, which converts a continuous function or signal into a discrete sequence. In units of cycles per second (Hz), its value is one-half of the sampling rate (samples per second). When the highest frequency (bandwidth) of a signal is less than the Nyquist frequency of the sampler, the resulting discrete-time sequence is said to be free of the distortion known as aliasing, and the corresponding sample-rate is said to be above the Nyquist rate for that particular signal.

By plotting the Fourier Transform of a signal, we can see that the Nyquist Frequency divides the signal to two mirrored parts. The left side of the plot (frequency 0 to Nyquist) is called the positive frequencies and the right side is the negative frequencies.

In the positive frequencies, increasing the frequency will result in the increase of frequency in sine waves of the Fourier Transform. But, passing the Nyquist frequency, this will get lower.

Correction Factor:

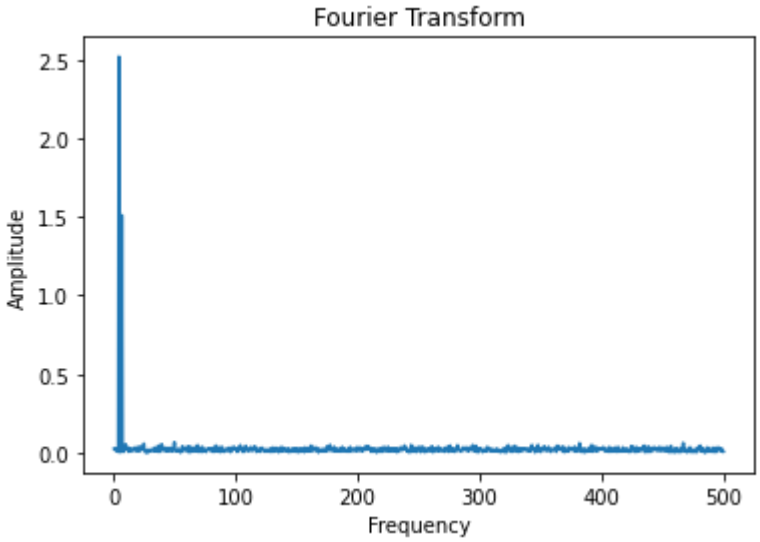
Correction factors for positive and negative frequencies(First factor):

Since our signal is divided to two parts, as mentioned in Nyquist Frequency, we'll need to multiply our calculation by 2.

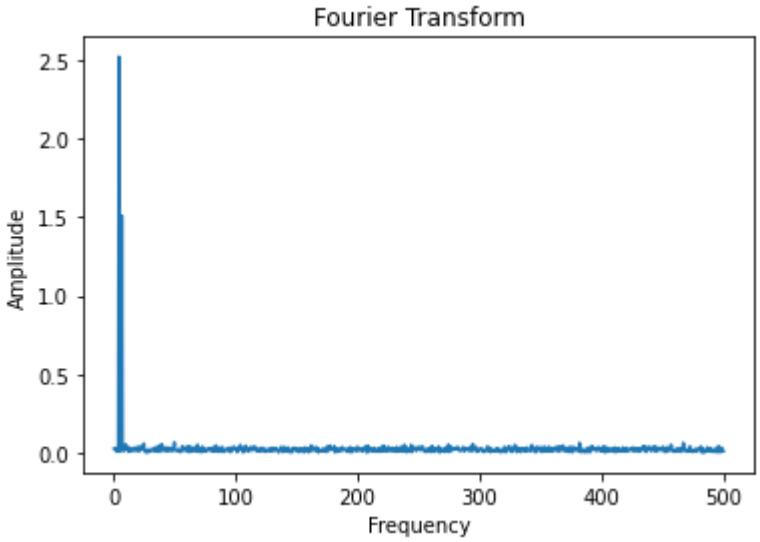
Correction factors for averaging in the Fourier Transform(Second factor): To make the average correct, we'll need to divide all calculation by size of our sampling(In this example, it's 2000)

Implementation:

2-2A:

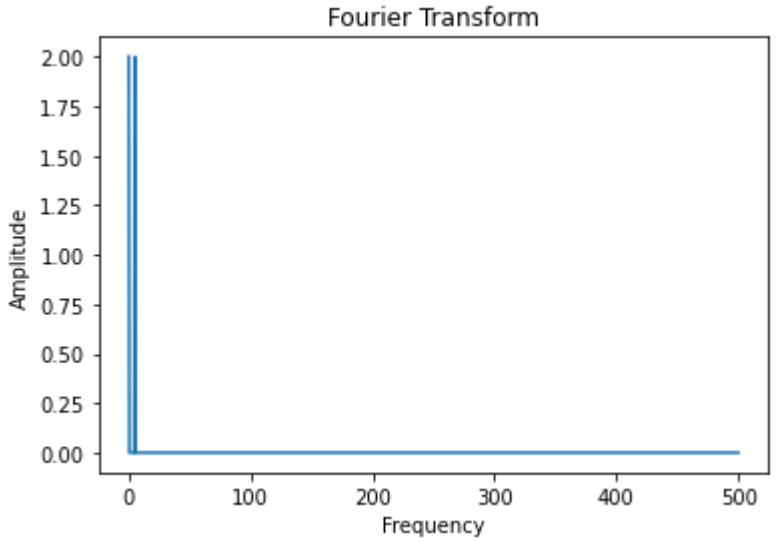


2-2B:

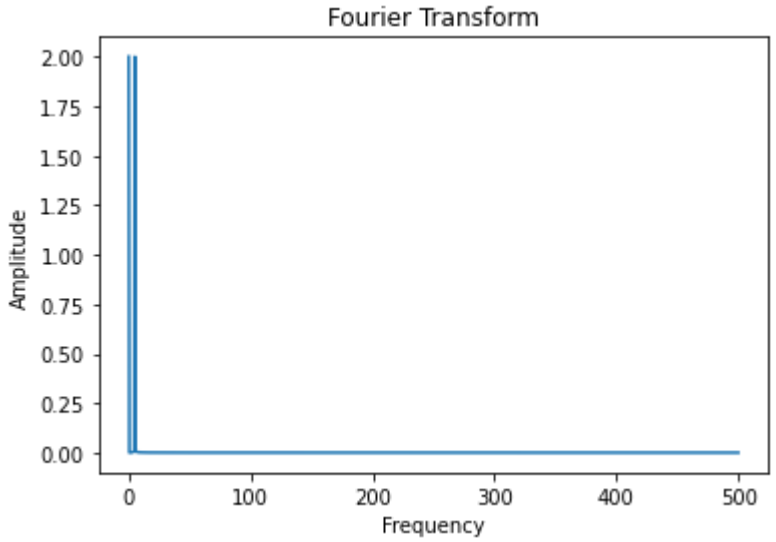


g(t) after correction:

Normal Method:



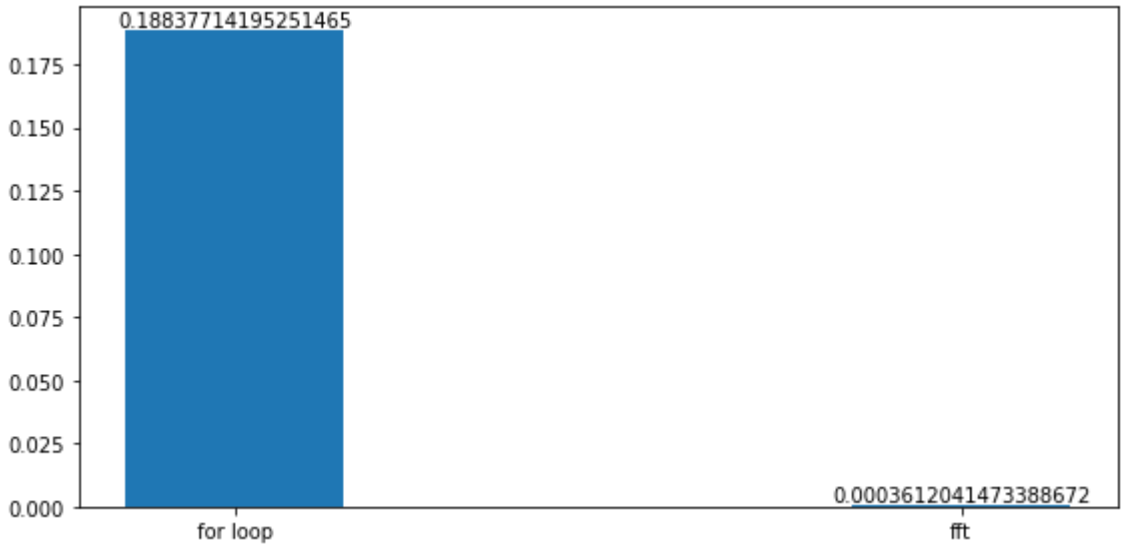
scipy.fft Method:



Before correction, we can see that the result is double the actual values This is because in the positive frequencies, we do not need to double the values i.e. we don't multiply the coefficients by the first factor. There are no negative frequencies in this part of the signal.

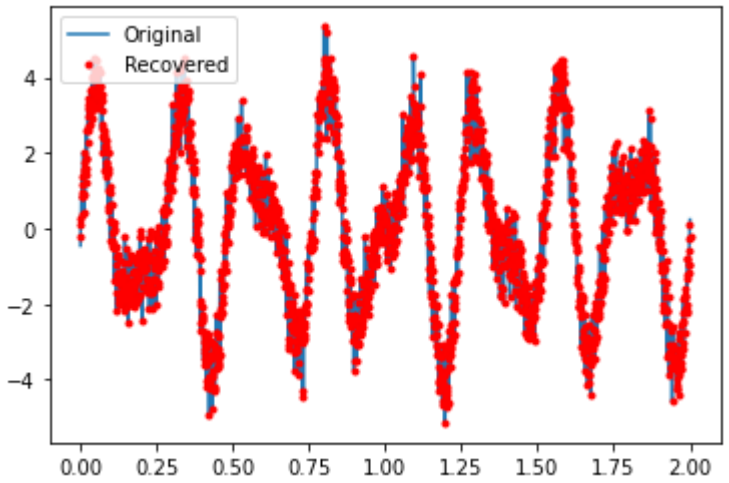
2-2C:

Differences:



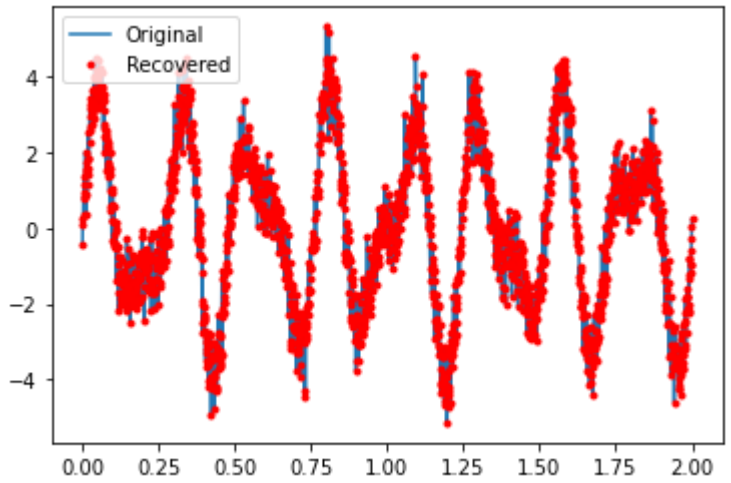
Question 3: Inverse Fast Fourier Transform(ifft)/ Discrete Inverse Fourier Transform

3A:



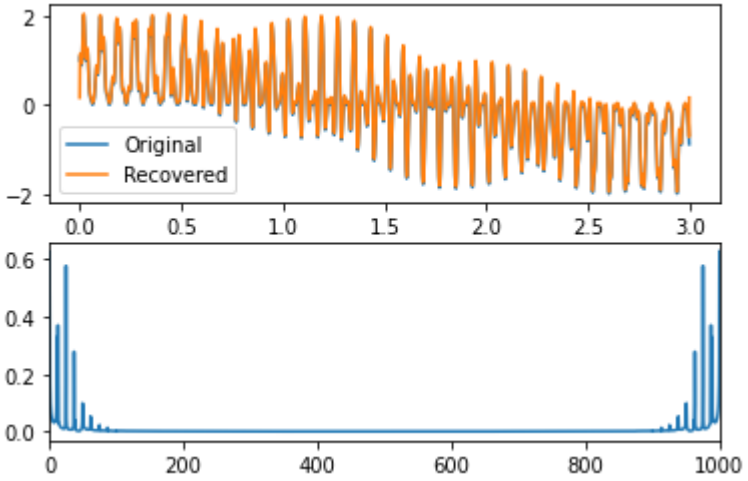
As you can see, most of the signal has been recovered and errors are because of computation limits. We can say that signal has been recovered successfully.

3B:

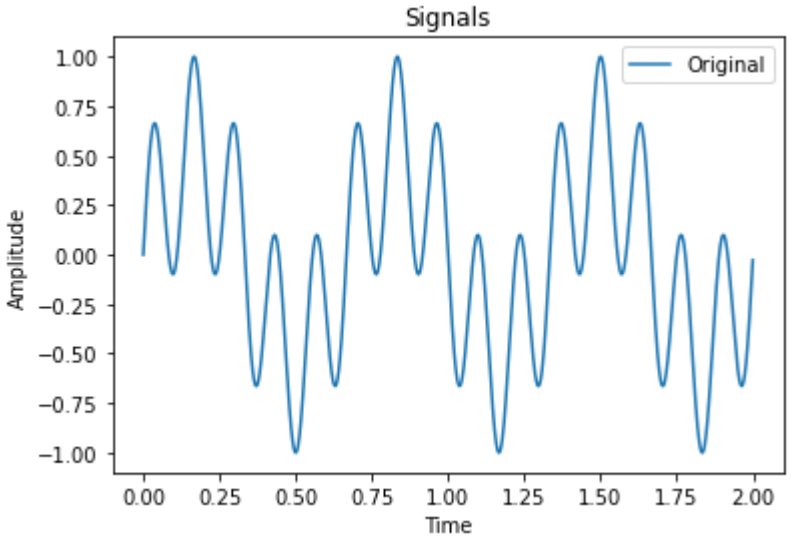


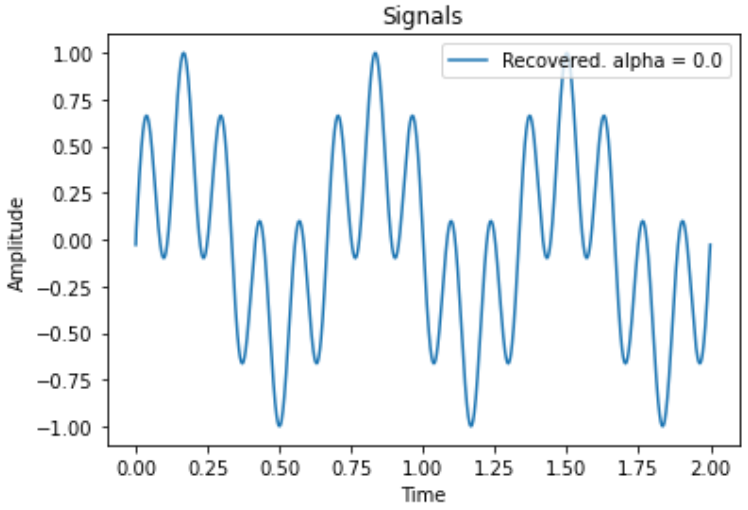
3C:

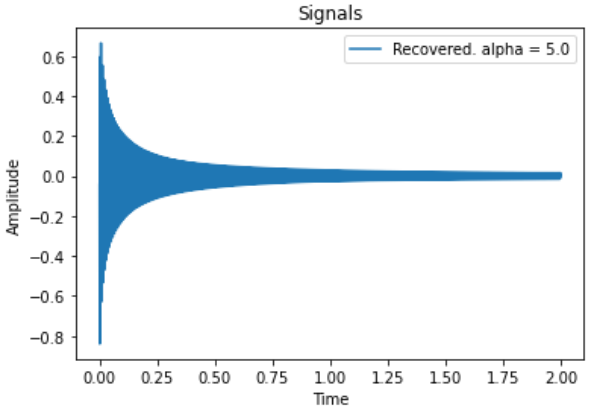
As you run the algorithm, you'll see that as coefficients in Fourier transform are calculated during the execution, the recovered signal is getting more identical to the original signal and when it's finished, the signal is almost recovered and there are some minor errors that are not that important.

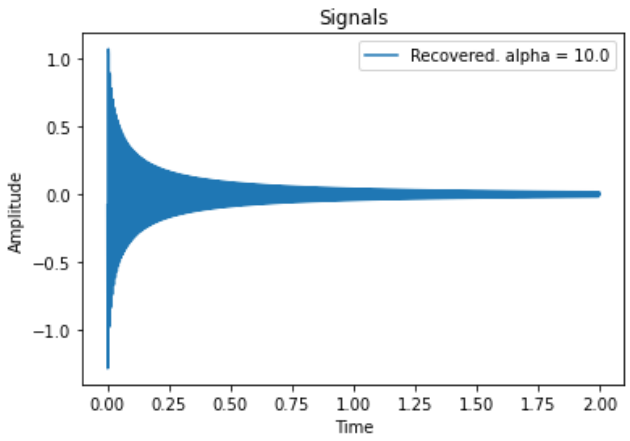


3D:

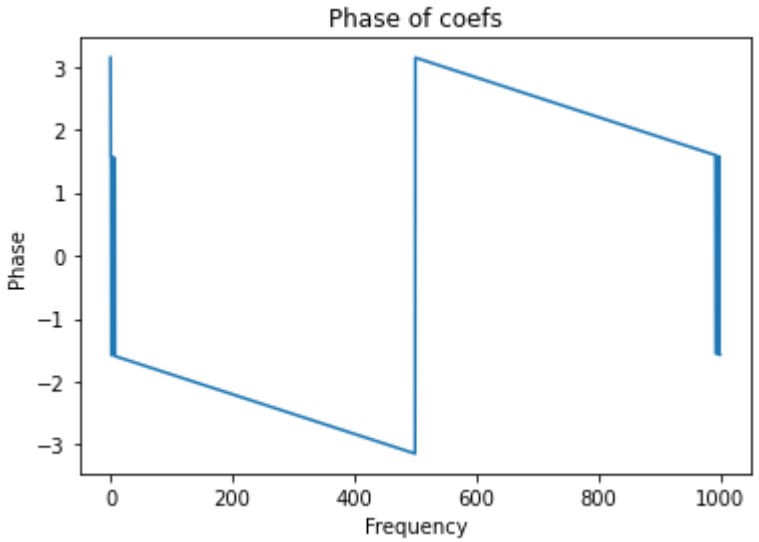

Signals
Original
Amplitude
Time
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

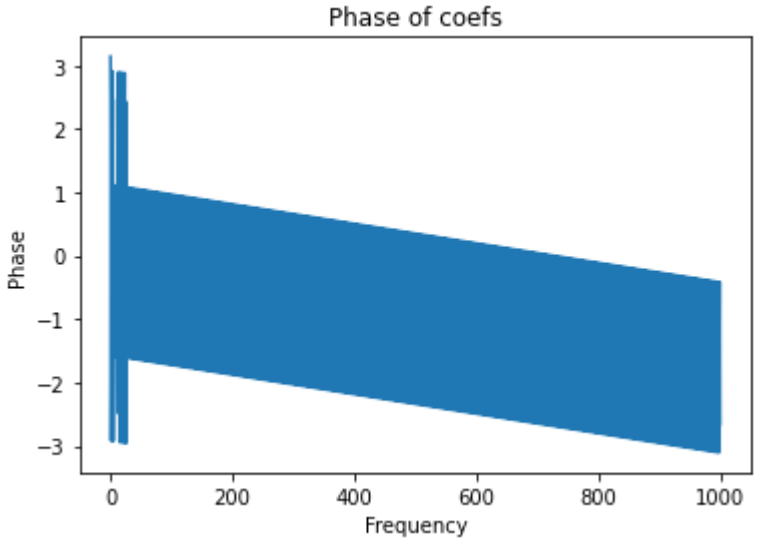

Signals
Recovered. alpha = 0.0
Amplitude
Time
1.00
0.75
0.50
0.25
0.00
−0.25
−0.50
−0.75
−1.00
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

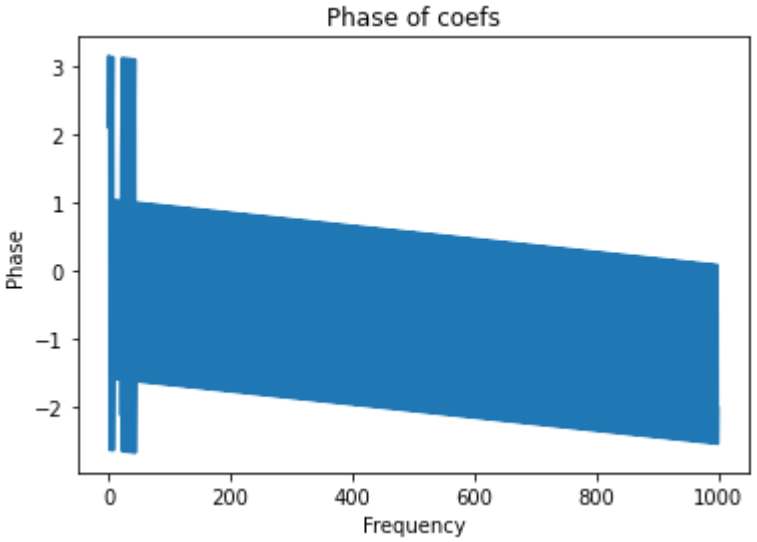

Signals
Recovered. alpha = 5.0
Amplitude
Time
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.8
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00



As you can see, the recovered signal with alpha = 0 is the most identical signal to the original signal and this is why we choose this method for calculating fft. This assumption can be accepted from phase plots below:

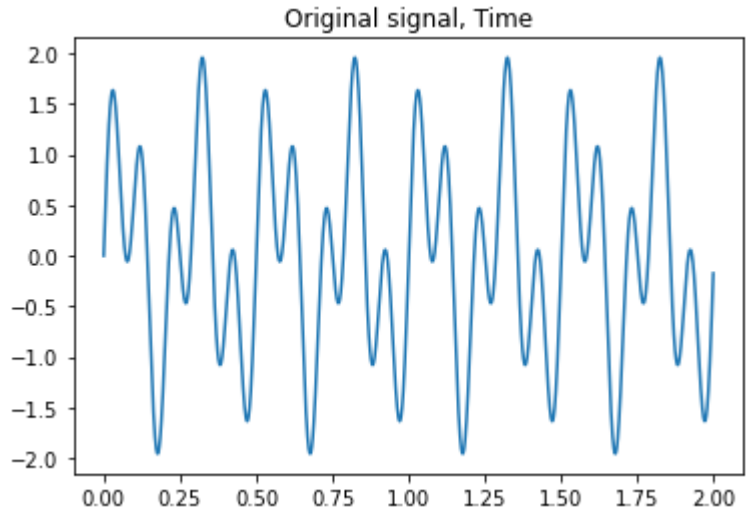


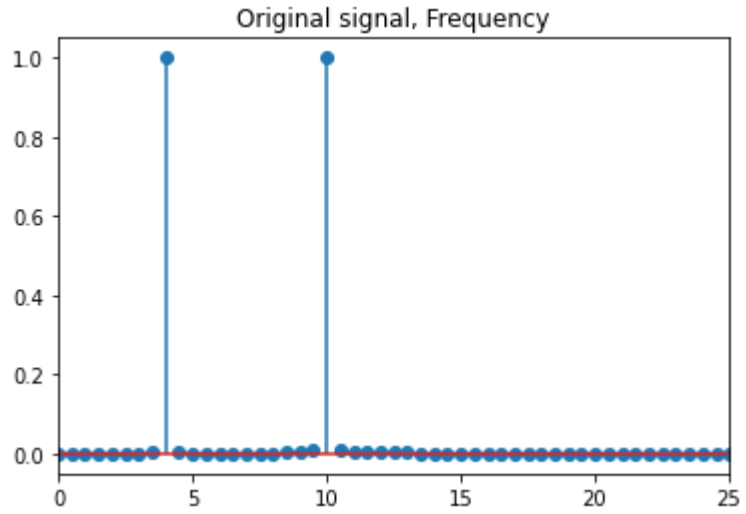


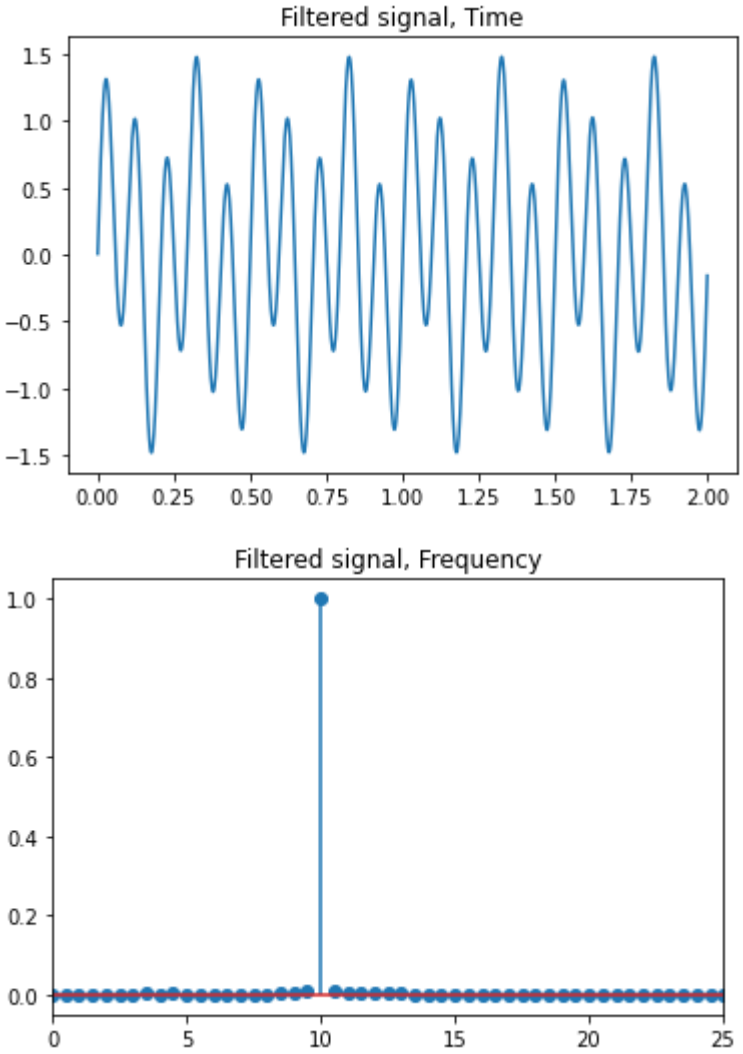


3E:

As you can see, after filtering freq = 4Hz from signal, the only frequency remaining is 10Hz:

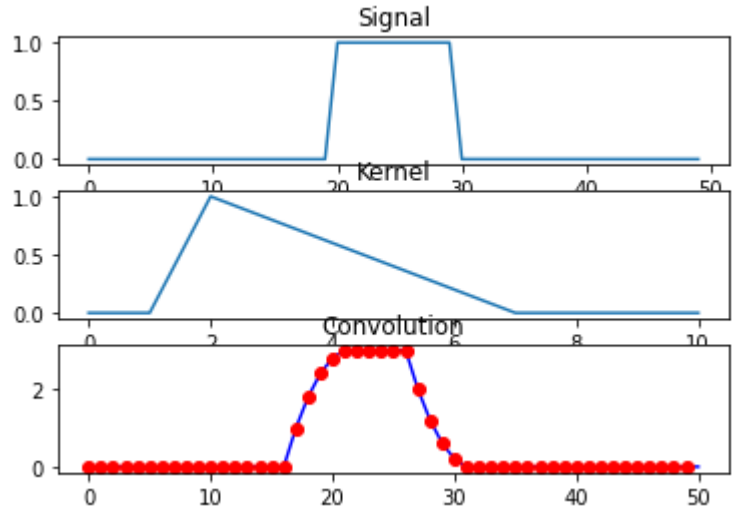




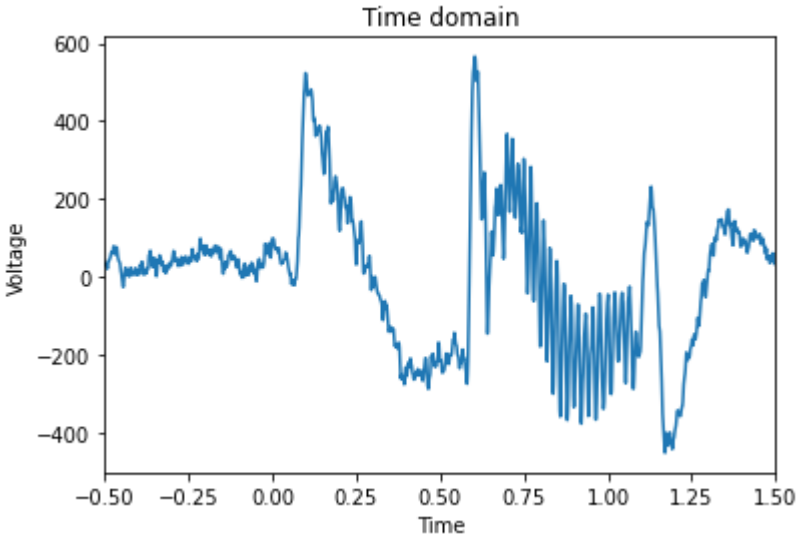


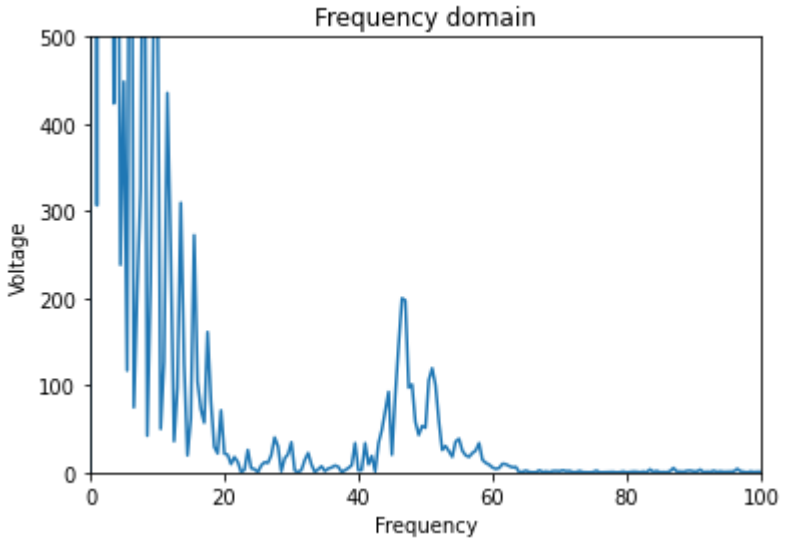
Question 4: Applictaions of Fourier Transform

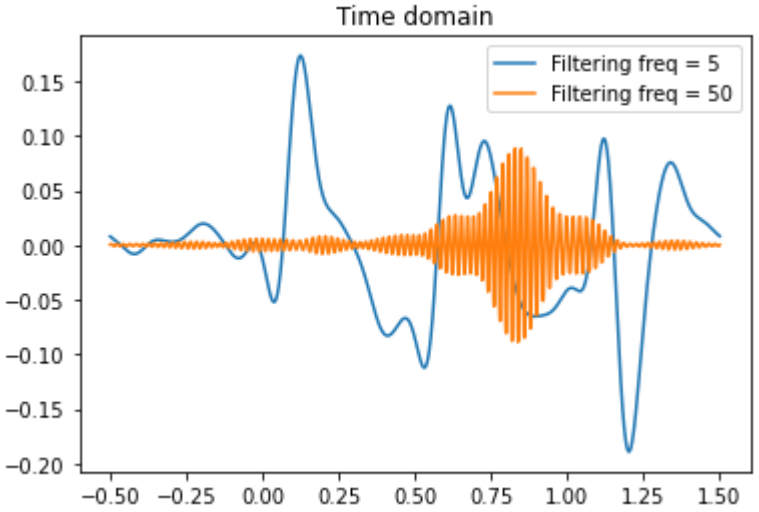
Convolution in Frequency-Domain:



Narrow band Temporal Filtering:







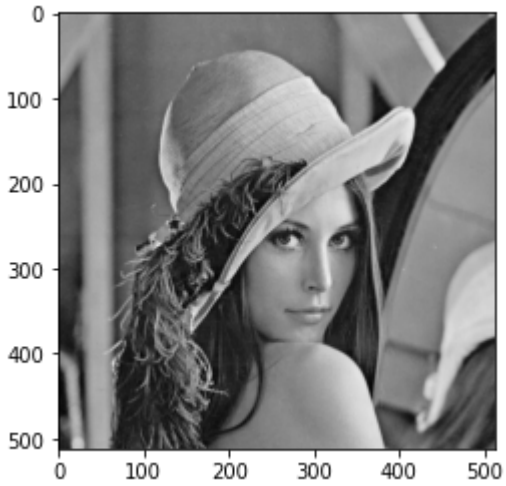
From the spectrum plot, we can say that these two filters for frequencies have done the correct filters. We have more powers in the frequencies less than 20 and between 40 and 60 and when we apply these filters, we capture frequencies with some limitations which

lead to the elimination of some frequencies, which according to chosen frequencies and spectrum above, will result in final filtered signals.

Image Filtering 2D:

Low Pass Filtering:

4-1A:



New shape of the image is (512, 512) after changing dimensions.

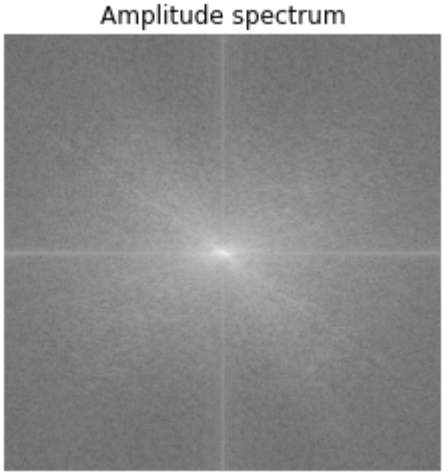
4-1B:

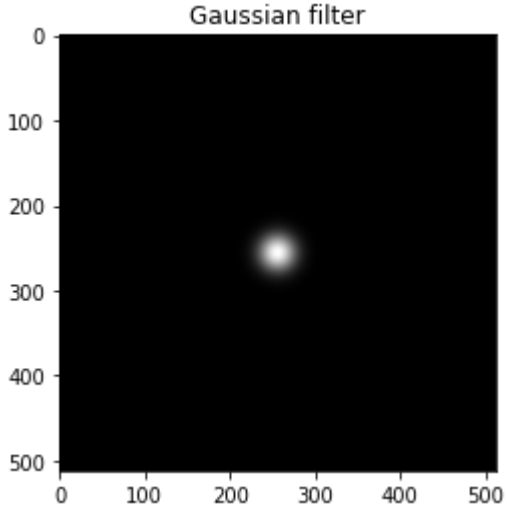
fft2: fft2 is just fftn with a different default for *axes*.

The output, analogously to fft, contains the term for zero frequency in the low-order corner of the transformed axes, the positive frequency terms in the first half of these axes, the term for the Nyquist frequency in the middle of the axes and the negative frequency terms in the second half of the axes, in order of decreasingly negative frequency. This function computes the N-D discrete Fourier Transform over any axes in an M-D array by means of the Fast Fourier Transform (FFT). By default, the transform is computed over the last two axes of the input array, i.e., a 2-dimensional FFT.

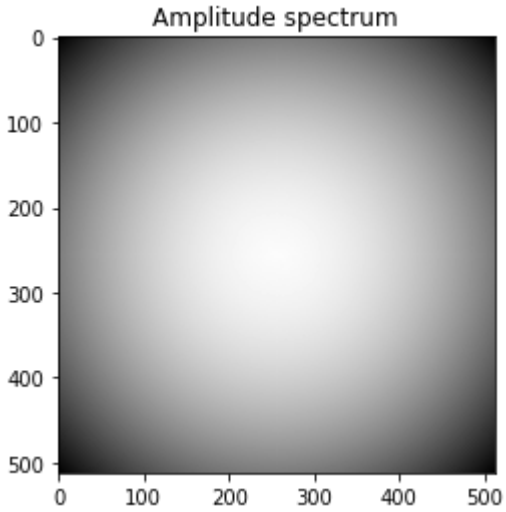
fftshift in 2D: Shift the zero-frequency component to the center of the spectrum. This function swaps half-spaces for all axes listed (defaults to all). Note that y[0] is the Nyquist component only if len(x) is even.

Initial spectrum:

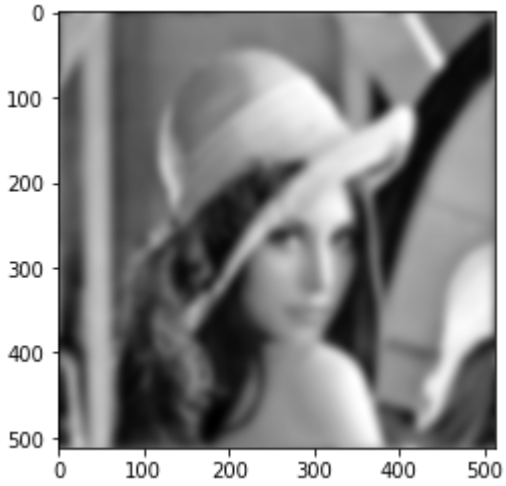




New spectrum:

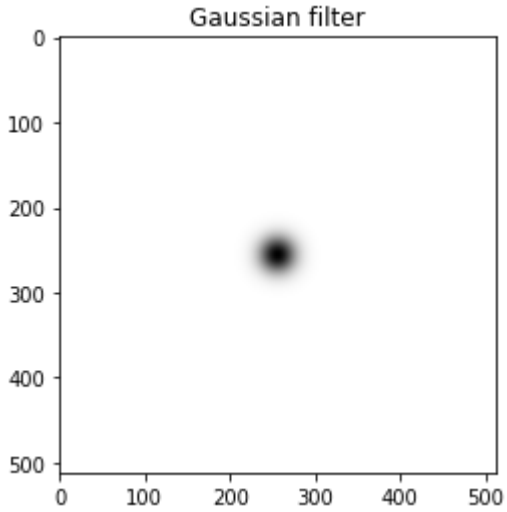


Blurred Image:

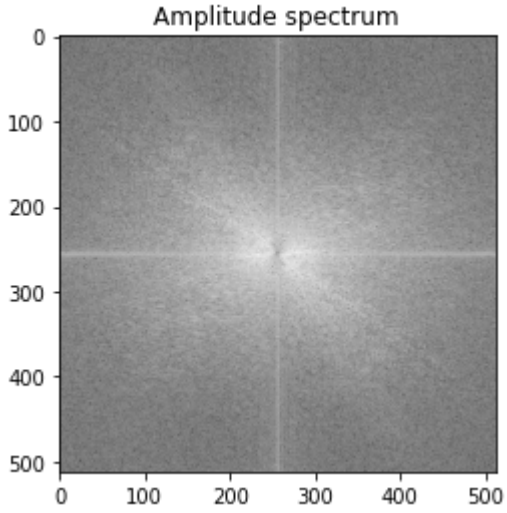


High Pass Filtering:

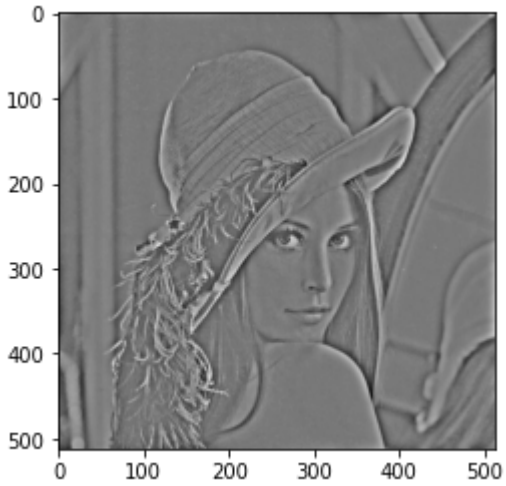
4-2A:



New spectrum:



Sharpened Image:



4-2B:

This filter sharpens the edges of the image. The Gaussian low pass filter attenuates frequency components that are further away from the center (W/2, H/2).The Gaussian high pass filter attenuates frequency components that are near the image center (W/2, H/2). We know that the borders and boundaries have higher frequencies, which help them pass the filter and as a result, we can see the borders clearly but without much details. High-pass and low-pass filters are also used in digital image processing to perform image modifications, enhancements, noise reduction, etc., using designs done in either the spatial domain or the frequency domain. The unsharp masking, or sharpening, operation used in image editing software is a high-boost filter, a generalization of high-pass.