PHYS 512 Problem Set 6

Teophile Lemay, 281081252

1

A convolution of two functions g and h is defined as the operation

$$(g * h)(t) = \int_{-\infty}^{\infty} d\tau g(t-\tau)h(\tau) \ .$$

A special case is the convolution with an impulse function $\delta(t-a)$, which is 1 at the impulse $t = a$ and 0 everywhere else.

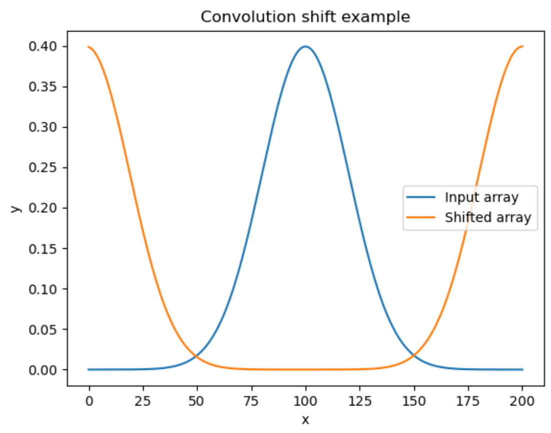
$$(f * \delta)(t) = \int_{-\infty}^{\infty} d\tau f(t-\tau)\delta(\tau - a)$$

Obviously, this integral is equal to 0 everywhere except at $\tau = a$, so

$$(f * \delta)(t) = \int_{-\infty}^{\infty} d\tau f(t-\tau)\delta(\tau - a) = f(t-a)$$

which results in shifting the array. I implemented this in code using Numpy's `np.convolve` function to convolve an input array with an impulse function. Generally, array shifting functions assume periodic boundary conditions. In keeping with this, my code performs the convolution of the shifted impulse with the input array concatenated with itself. The second half of the convolution output is returned as the shifted array. Figure 1 shows the result of shifting an array containing a Gaussian by half its length. Code for this question is in `Q1_convolution_shift.py`.

Figure 1: Convolution shift of a Gaussian

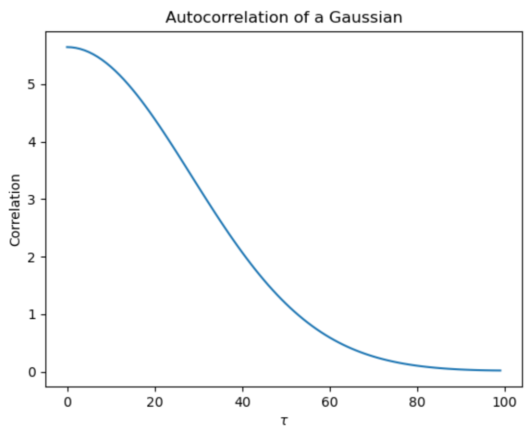


2

2.1 a)

My correlation function correlates two input arrays by taking the DFT for each and returning the inverse DFT of the product of the first transformed array with the conjugate of the second array, as per the definition of the correlation by Fourier transform. For simplicity, I only return the correlation for positive lengths. Figure 2 shows the autocorrelation (correlation with itself) of the same Gaussian as used in question 1. The code for this question is in Q2a_correlation_function.py.

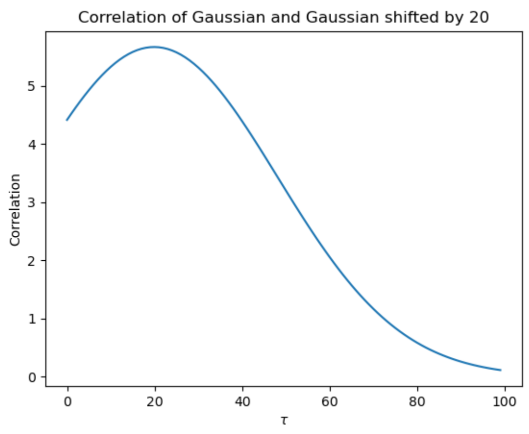
Figure 2: Correlation of a Gaussian with itself



2.2 b

Figure 3 shows the correlation of the Gaussian with itself shifted by 20. Both operations were performed using the functions created in the parts above. As expected, the correlation function has a peak at $\tau = 20$.

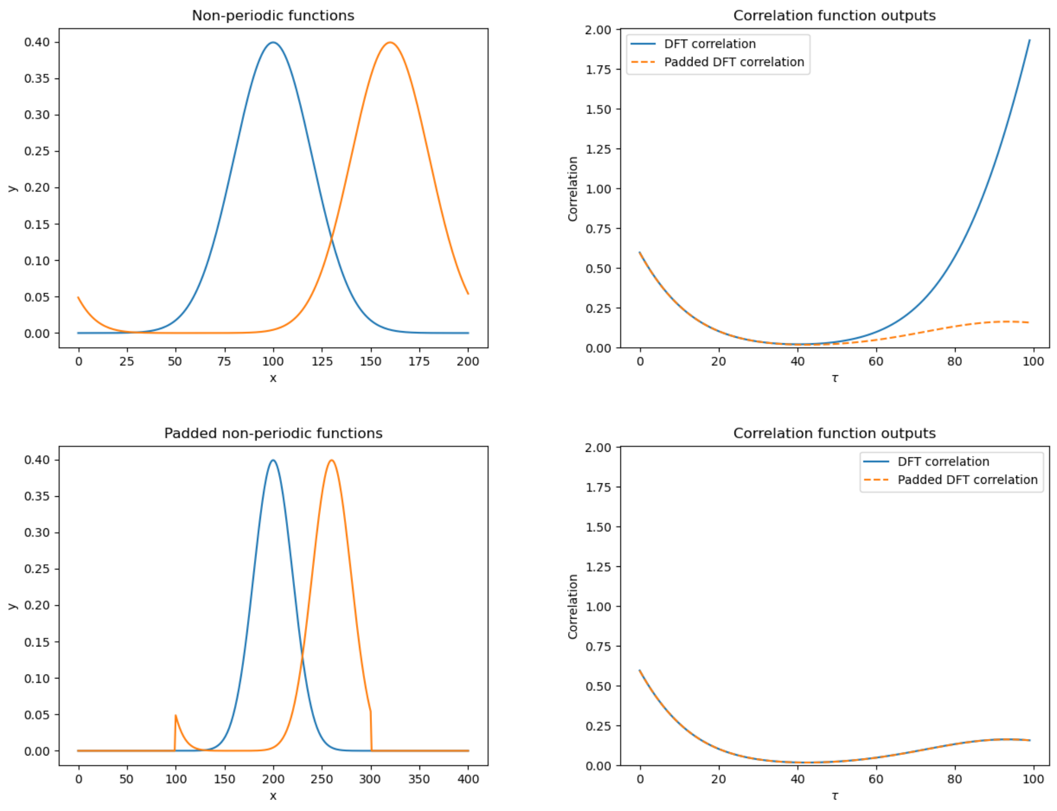
Figure 3: Correlation of Gaussian and shifted Gaussian



3

In order to prevent the DFT correlation from including the effects of periodic boundary conditions, my function padds the input arrays with zeros on both sides before performing the convolution. After performing the correlation, only positive τ values from 0 to half the length of the arrays are returned. In order to compare this function with my DFT correlation function from question 2, I performed the correlation of a Gaussian centered at the middle of an array with the same Gaussian shifted to the right. The top two plots in Figure 4 show the functions and the results of the correlations. In this case, there is a clear difference between both results, and the plain DFT correlation function shows high correlation for higher τ values that can only exist if both functions have periodic boundary conditions. In contrast, the bottom row of plots shows the functions with an added padding of zeros on both sides. This padding removes the effect of periodic boundary conditions for low τ values since the offset in the correlation does not reach high enough values for the periodic boundary conditions to have any effect. In this case, both the plain, and padded DFT correlation functions give the same output. From these results, I can conclude that my padded DFT correlation function is properly correlating its inputs while removing the periodic boundary condition effects of a DFT correlation. Code for this question can be found in `Q3_non_periodic.py`.

Figure 4: Comparing padded and non-padded DFT correlation functions



4

4.1 a)

W.T.S.

$$\sum_{x=0}^{N-1} e^{\frac{-2\pi ikx}{N}} = \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)}$$

Rewriting the LHS, this becomes the sum of a geometric series:

$$\sum_{x=0}^{N-1} \left(e^{\frac{-2\pi ik}{N}}\right)^x = \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)}\ .$$

This sum has a known result:

$$\sum_{k=0}^{n-1} r^k = \frac{1-r^n}{1-r}$$

for $r \neq 1$. Therefore

$$\sum_{x=0}^{N-1} \left(e^{\frac{-2\pi ik}{N}}\right)^x = \frac{1-\exp(-2\pi ik/N)^N}{1-\exp(-2\pi ik/N)} = \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)} \ .$$

4.2 b)

In the limit $k \to 0$, the sum becomes

$$\lim_{k\to 0} \sum_{x=0}^{N-1} \left(e^{\frac{-2\pi ik}{N}}\right)^x = \sum_{x=0}^{N-1} \left(e^{\frac{-2\pi i\cdot 0}{N}}\right)^x = \sum_{x=0}^{N-1} (1)^x \ .$$

The sum of a geometric series also has a known result for this case:

$$sum_{k=0}^{n-1} r^k = n \quad \text{for} \quad r = 1 \ .$$

Therefore, in the case $k \to 0$,

$$\sum_{x=0}^{N-1} \left(e^{\frac{-2\pi ik}{N}}\right)^x = N \ .$$

If $k \neq 0$, but is also not an integer multiple of N, the sum of geometric series identity still holds so

$$\sum_{x=0}^{N-1} \left(e^{\frac{-2\pi ik}{N}}\right)^x = \frac{1-\exp(-2\pi ik/N)^N}{1-\exp(-2\pi ik/N)} = \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)} \ .$$

The complex exponentials can be re-written according to Euler's formula to give

$$\frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)} = \frac{1-(\cos(-2\pi k)+i\sin(-2\pi k))}{1-\left(\cos\left(\frac{-2\pi k}{N}\right)+i\sin\left(\frac{-2\pi k}{N}\right)\right)}$$
$$= \frac{1-(\cos(2\pi k)-i\sin(2\pi k))}{1-\left(\cos\left(\frac{2\pi k}{N}\right)-i\sin\left(\frac{2\pi k}{N}\right)\right)} \ .$$

For integer k, $\sin(2\pi k) = 0$ and $\cos(2\pi k) = 1$ so the numerator is 0 for any integer k. Furthermore, if k is not a multiple of N, then $\cos(2\pi k/N) < 1$ so the denominator is non-zero (and complex for $k \neq N/2$). Thus we have

$$\sum_{x=0}^{N-1} \left(e^{\frac{-2\pi ik}{N}}\right)^x = \frac{0}{z} = 0$$

where z is some finite complex number.

4.3 c)

Using Euler's formula again, I can write a non-integer sine wave as

$$\sin ax = \frac{e^{iax}-e^{-iax}}{2i}$$

where a is not an integer multiple of 2π. The DFT of this sine wave is

$$\sum_{x=0}^{N-1} e^{\frac{-2\pi ikx}{N}} \sin ax = \sum_{x=0}^{N-1} e^{\frac{-2\pi ikx}{N}} \frac{e^{iax}-e^{-iax}}{2i} = \frac{1}{2i}\left[\sum_{x=0}^{N-1} e^{\frac{-2\pi ikx}{N}+iax} - \sum_{x=0}^{N-1} e^{\frac{-2\pi ikx}{N}-iax}\right]$$
$$= \frac{1}{2i}\left[\sum_{x=0}^{N-1}\left(e^{\frac{-2\pi ik}{N}+ia}\right)^x - \sum_{x=0}^{N-1}\left(e^{\frac{-2\pi ik}{N}-ia}\right)^x\right] = \frac{1}{2i}\left[\frac{1-e^{-2\pi ik+iaN}}{1-e^{\frac{-2\pi ik}{N}+ia}} - \frac{1-e^{-2\pi ik-iaN}}{1-e^{\frac{-2\pi ik}{N}-ia}}\right] \ .$$

With Euler's formula,

$$= \frac{1}{2i}\left[\frac{1-(\cos(-2\pi k+aN)+i\sin(-2\pi k+aN))}{1-\left(\cos\left(\frac{-2\pi k}{N}+a\right)+i\sin\left(\frac{-2\pi k}{N}+a\right)\right)} - \frac{1-(\cos(-2\pi k-aN)+i\sin(-2\pi k-aN))}{1-\left(\cos\left(\frac{-2\pi k}{N}-a\right)+i\sin\left(\frac{-2\pi k}{N}-a\right)\right)}\right].$$

Trigonometric functions are periodic over 2π so the phase shifts of $-2\pi k$ in the denominator can be ignored.

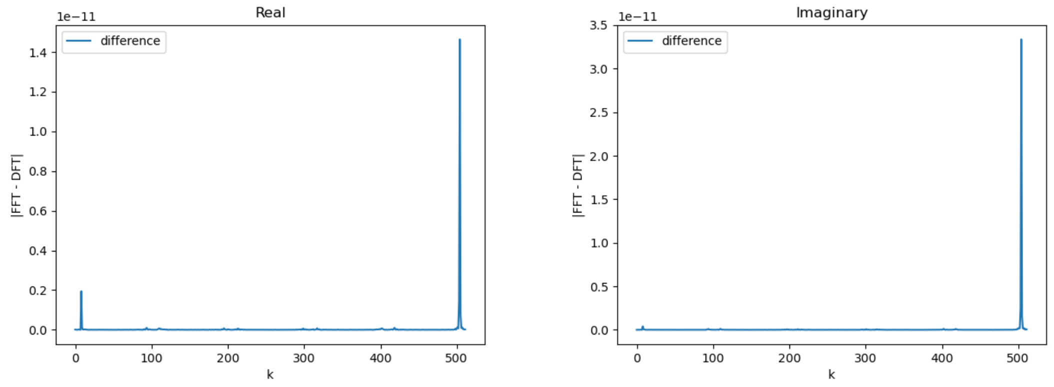
$$\sum_{x=0}^{N-1} e^{\frac{-2\pi ikx}{N}}\sin ax = \frac{1}{2i}\left[\frac{1-(\cos(aN)+i\sin(aN))}{1-\left(\cos\left(\frac{-2\pi k}{N}+a\right)+i\sin\left(\frac{-2\pi k}{N}+a\right)\right)} - \frac{1-(\cos(aN)-i\sin(aN))}{1-\left(\cos\left(\frac{2\pi k}{N}+a\right)-i\sin\left(\frac{2\pi k}{N}+a\right)\right)}\right].$$

I compared this result to the Numpy's FFT implementation on the function

$$f(x) = \sin(0.1x)$$

evaluated over the integers from 0 to 1023 (inclusive). Figure 5 shows the real and imaginary parts of the difference of both transforms. The scale of the differences is on the order of 10^{-11} which is larger than machine precision but still an extremely close match between the FFT and the analytical DFT. Figure 6 shows the power spectrum of the sine wave (as computed from the analytical DFT). While it has a sharp peak, there is still some spectral leakage as the DFT is not able to perfectly capture the frequency of the function. Code for this question is in `Q4_analytic_DFT.py`.

Figure 5: Difference of Analytical DFT and FFT transforms



4.4 d)

I applied the Hann window to the same non-integer since function used in part c), and evaluated the FFT of the windowed and non-windowed function using Numpy's FFT function. As shown in figure 7, the peak of the power spectrum for the windowed function is narrower with much smaller "tails" on each side due to spectral leakage. Code for this question is in `Q4_hann_window.py`.

Figure 6: Non-integer sine wave power spectrum

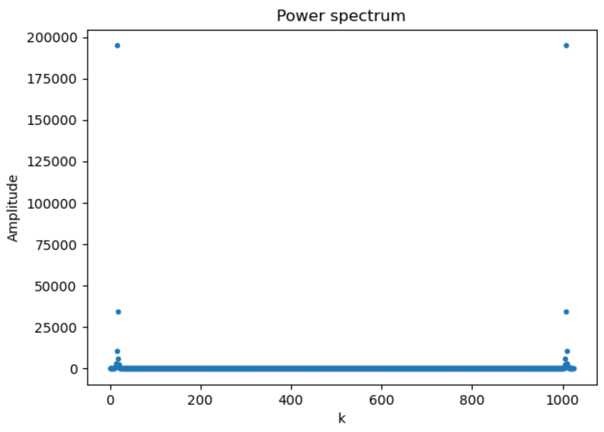
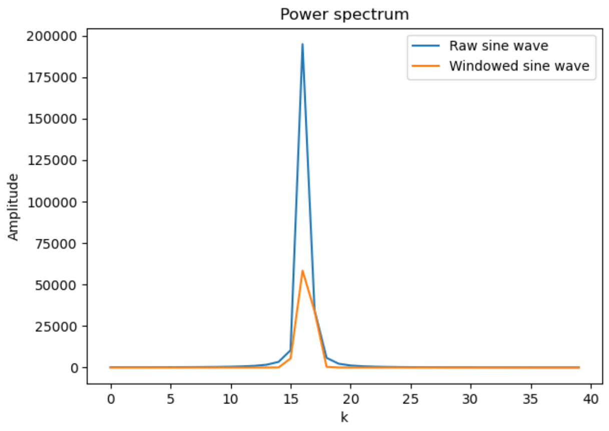


Figure 7: Power spectrums with and without windowing


Power spectrum
Raw sine wave
Windowed sine wave
Amplitude
k