Phys 512 Assignment 5

Ian Hendricksen

October 28, 2021

1. A convolution of a function with a delta function simply shifts the function to where the delta function is centered:

$$f(x) \circledast \delta(x-\gamma) = f(x-\gamma). \tag{1}$$

My function `shift(y, dy)` takes an array y and shifts it by dy. It constructs a delta function such that an array of zeros with the same length as y has a value of 1 at index $dy-1$ (the -1 accounts for python indexing from 0). Plotted in Figure 1 from -5 to 5 is `gauss_shift.png`, which is a Gaussian centered at -2.5 that is shifted by half the array length. The array length in this plot is 10000, so we shift the Gaussian by 5000 (note that the 5000 refers to indices), which corresponds to shifting the Gaussian from $x=-2.5$ to $x=2.5$.

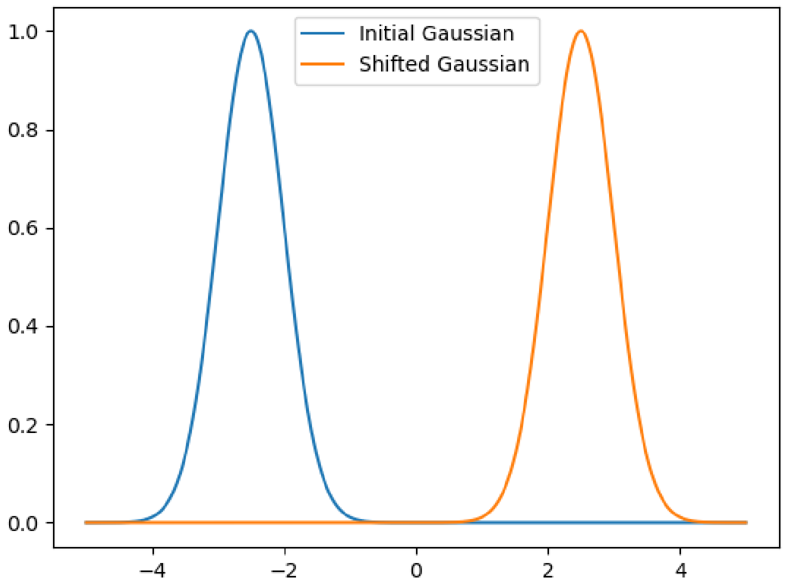

Figure 1: A convolution of some function with a delta function returns the same function shifted to the center of the delta function.

2. We may write the correlation function as

$$f \star g = \mathcal{F}^{-1}\left(\mathcal{F}(f)\mathcal{F}(g)^*\right), \tag{2}$$

which is computed using my function `corr(f,g)`. Using this, we compute the correlation of a Gaussian (centered at 0 with amplitude 0.1) with itself, which is plotted in Figure 2 (also saved as `gauss_corr.png`). I set the amplitude to be 0.1 so we can see the effect of the correlation, namely that the amplitude has increased by a factor of ∼17.5, and that the width of the Gaussian has increased.

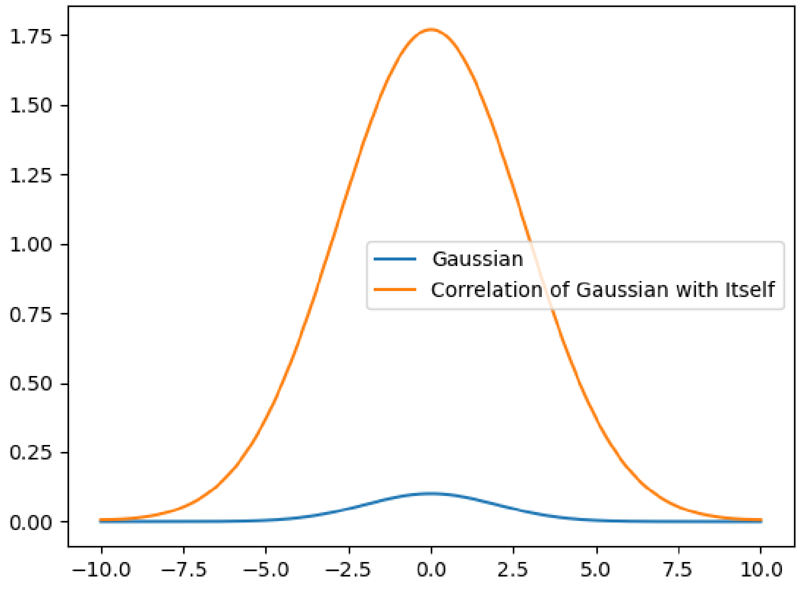

Figure 2: A Gaussian, centered at 0 and having amplitude 0.1, correlated with itself.

3. Here, we begin with a Gaussian centered at 0 and with amplitude 0.1. We shift it by some arbitrary amount using `shift(y, dy)`, and compute the correlation of this shifted Gaussian with itself. The result is plotted in Figure 3 showing the original, shifted, and correlated Gaussians. What is interesting to note is that the shift has no impact whatsoever on the correlation; the shifted Gaussian correlated with itself always wraps around in the manner we see in Figure 3. I was certainly surprised to see this effect, BUTTTTTTTTTTTTTTTTTTTTTTTTTTTTT

4. Here, I have written a function `conv_safe(f, g, N_zeros)` that pads the arrays `f` and `g` with zeros based on their relative lengths to avoid the issue of wrap-arounds when computing a convolution (similar to what we had seen in Figure 3). It begins by comparing the lengths of each array, and adds `N_zeros` (i.e. some user-specified number of zeros) to the end of the array, and then adds `N_zeros` plus a number of zeros equal to the difference in lengths of the two arrays, such that the "padded" `f` and `g` have equal length (which means that the output array will have a length equal to that of the longer array plus the number of zeros specified by the user). If the arrays are of equal size, `N_zeros` are added to the ends of both arrays. It then computes the FFT of the padded arrays, and the IFFT of the multiplication of the two FFT's to return the convolved function. I have convolved a Gaussian with amplitude 0.1 with itself using `conv_safe`, which is displayed in Figure 4. In this instance we see that the convolved Gaussian does not wrap around, and it is fully continuous (within the x-axis limits) as opposed to that seen in Figure 3, where some wrapping is evident.

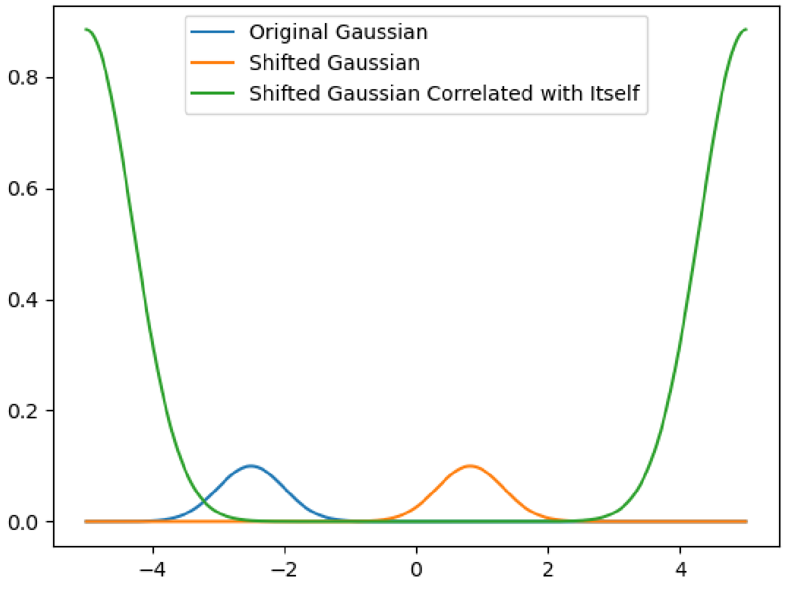

Figure 3: A Gaussian in its original, shifted, and shifted correlated with itself (i.e. a correlation of the same shifted Gaussian) forms.

5. (a) To prove that

$$\sum_{x=0}^{N-1} e^{-2\pi ikx/N} = \frac{1-e^{-2\pi ik}}{1-e^{-2\pi ik/N}}, \tag{3}$$

we can begin by stating that a geometric series is given by

$$\sum_{x=0}^{N-1} r^x = \frac{1-r^N}{1-r}. \tag{4}$$

If we set $r = e^{-2\pi ik/N}$, we can simply substitute r into Equation 4:

$$\sum_{x=0}^{N-1} \left(e^{-2\pi ik/N}\right)^x = \frac{1-e^{(-2\pi ik/N)N}}{1-e^{-2\pi ik/N}} = \frac{1-e^{-2\pi ik}}{1-e^{-2\pi ik/N}}. \tag{5}$$

(b) We can show that the above approaches N as k approaches 0 by taking the limit $k \to 0$:

$$\lim_{k\to 0} \sum_{x=0}^{N-1} e^{-2\pi ikx/N} \to \sum_{x=0}^{N-1} e^0 \to \sum_{x=0}^{N-1} 1 \to N. \tag{6}$$

Further, this can be shown to go to 0 for any k that is not a multiple of N:

$$nnskfdjndskfndskfn \tag{7}$$

(c) The DFT of a sin wave $f(x) = \sin(2\pi kx/N)$ may be written as

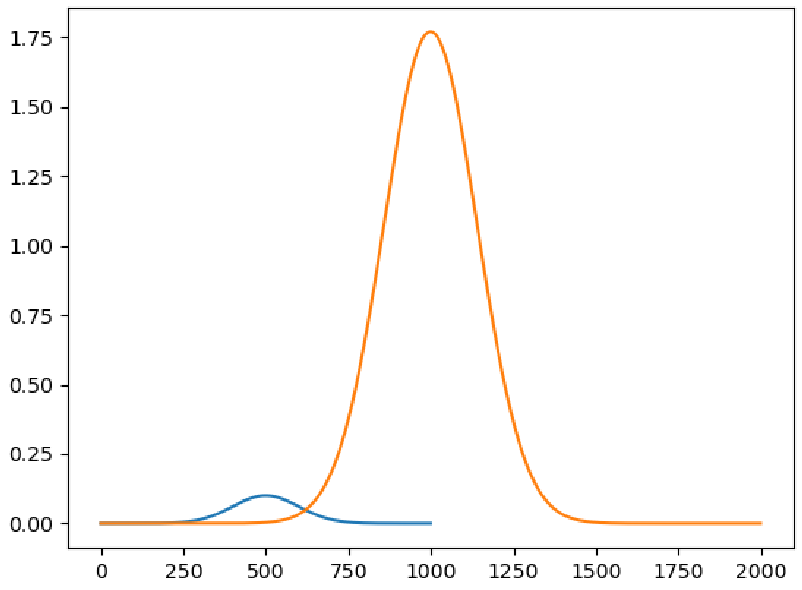

Figure 4: A "safe" convolution of a Gaussian with itself. There is no wrap-around issue occurring here.

$$F(k) = \sum_{x=0}^{N-1} f(x) e^{-2\pi i k' x/N} = \sum_{x=0}^{N-1} \sin(2\pi k x/N) e^{-2\pi i k' x/N}, \tag{8}$$

where k may be some arbitrary (non-integer) number. Our sin function may also be written in exponential form as

$$sin(2\pi k x/N) = \frac{1}{2i}\left(e^{2\pi i k x/N} - e^{-2\pi i k x/N}\right). \tag{9}$$

We can replace this form into Equation 8:

$$F(k) = \sum_{x=0}^{N-1} \frac{1}{2i}\left(e^{2\pi i k x/N} - e^{-2\pi i k x/N}\right) e^{-2\pi i k' x/N}, \tag{10}$$

which allows us to break up the sum as

$$F(k) = \frac{1}{2i}\left(\sum_{x=0}^{N-1} e^{2\pi i k x/N} e^{-2\pi i k' x/N} - \sum_{x=0}^{N-1} e^{-2\pi i k x/N} e^{-2\pi i k' x/N}\right). \tag{11}$$

This further reduces to

$$F(k) = \frac{1}{2i}\left(\sum_{x=0}^{N-1} e^{2\pi i (k-k') x/N} - \sum_{x=0}^{N-1} e^{-2\pi i (k+k') x/N}\right), \tag{12}$$

and following the proof in (a), we may state

$$F(k) = \frac{1}{2i}\left(\frac{1 - e^{2\pi i (k-k')}}{1 - e^{2\pi i (k-k')/N}} - \frac{1 - e^{-2\pi i (k+k')}}{1 - e^{-2\pi i (k+k')/N}}\right). \tag{13}$$

We can, of course, "simplify" (ostensibly) this a bit further so it works nicely in Python by writing this in sines and cosines, but it's a little bulky, so I'll just leave that in the code.

I have plotted

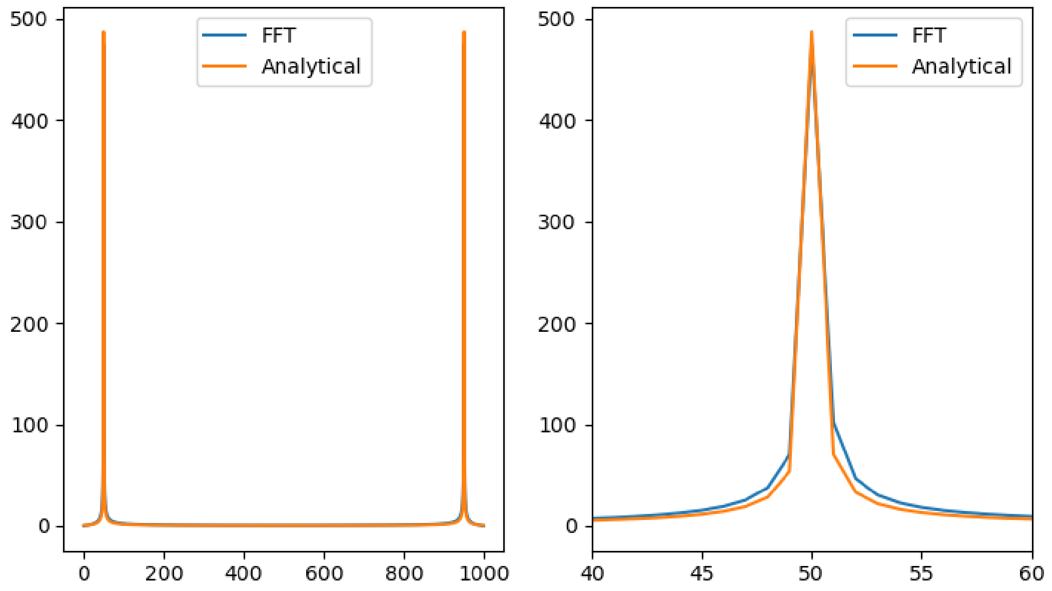

Figure 5: FFT of a sin function with non-integer k, using both NumPy's FFT and the analytical DFT of Equation 13. I have included a zoomed-in plot on the right so that the spectral leakage is apparent.

(d)

(e) To show that the FT of the window is $[N/2, -N/4, \ldots, -N/4]$, each time the script is run, a text file called `cos_wind_ft.txt` is written that compares the first two and last elements of the FT of the window function with the values listed above, and prints the rest of the FT to see if the values are all close to 0 as expected. The current run of the script appears to suggest that it is indeed the case that the FT of the window function agrees with the values I listed above (note that I have taken the absolute value, so the negatives are lost). The text file may be found under the same `ps5` folder.

6.