Phys 512 Assignment 5

Ian Hendricksen

October 29, 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 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 $x = 2.5$.

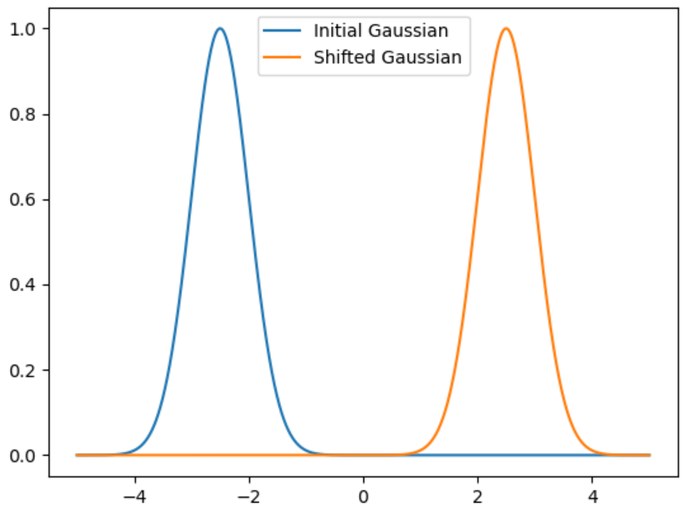

Figure 1: A convolution of some function with a delta function returns the same function shifted.

2. We may write the correlation function as $f \star g = \mathcal{F}^{-1}\left(\mathcal{F}(f)\mathcal{F}(g)^*\right)$, 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 $\sim$17.5, and that the width of the Gaussian has increased.

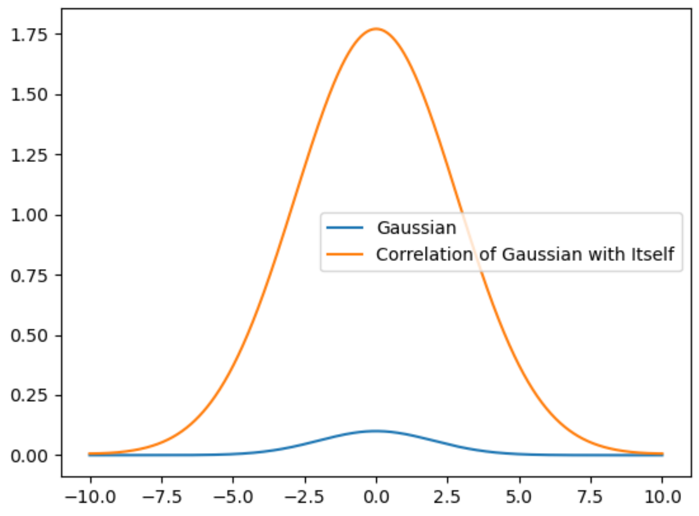

Figure 2: A Gaussian, centered at 0 and having amplitude 0.1, correlated with itself.

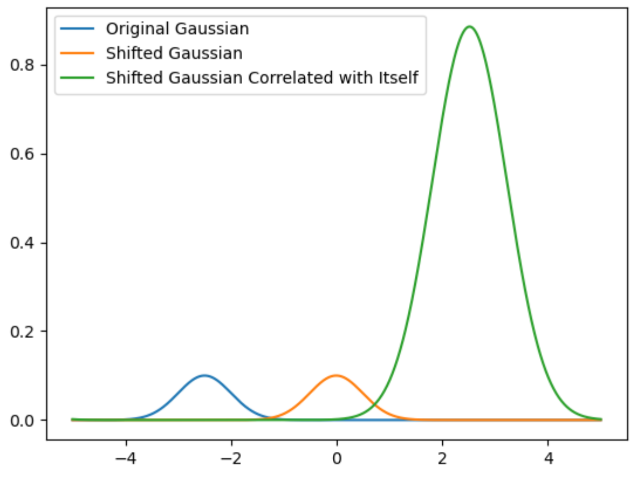

Figure 3: The correlation of a Gaussian and a shifted Gaussian is a shifted Gaussian.

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 the original unshifted Gaussian. The result is plotted in Figure 3 showing the original, shifted, and correlated Gaussians. This demonstrates that a correlation of a function and the same function, but with a shift applied, results in the correlated function being shifted. It also shifts in the same direction as the original shift, and by the same amount; this makes sense, since if we were to correlate a Gaussian with itself, the correlated function would be a Gaussian centered at 0 (as in Figure 2), so if we correlate a Gaussian with a shifted Gaussian, the output should also be shifted from 0.

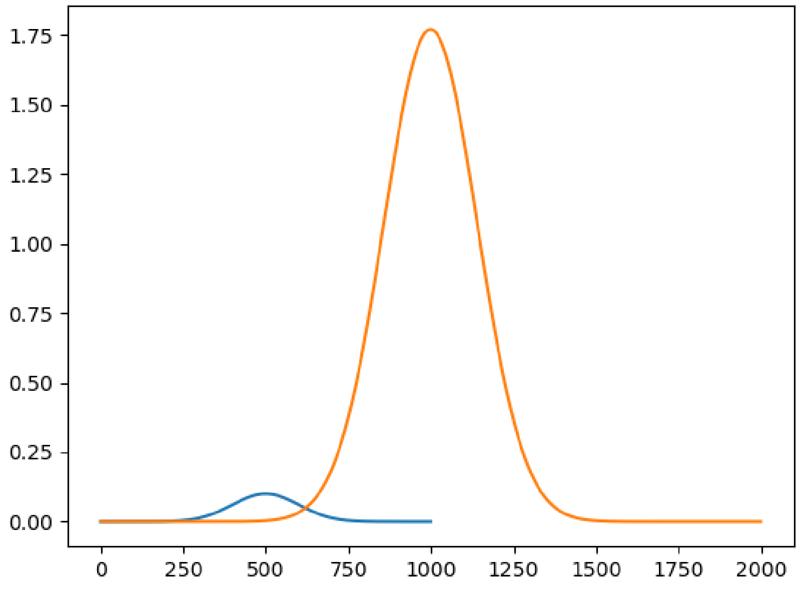

Figure 4: A "safe" convolution of a Gaussian with itself. There is no wrap-around issue occurring here.

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.

5. (a) To prove that

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

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

$$\sum_{x=0}^{N-1} r^x = \frac{1-r^N}{1-r}. \tag{3}$$

If we set $r = e^{-2\pi ik/N}$, we can simply substitute r into Equation 3:

$$\sum_{x=0}^{N-1} \left(e^{-2\pi ik/N}\right)^x = \frac{1-e^{(-2\pi ik/N)N}}{1-e^{-2\pi ik/N}} = \frac{1-e^{-2\pi ik}}{1-e^{-2\pi ik/N}}. \tag{4}$$

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

$$\lim_{k\to 0} \sum_{x=0}^{N-1} e^{-2\pi ikx/N} \to \sum_{x=0}^{N-1} e^0 \to \sum_{x=0}^{N-1} 1 \to N. \tag{5}$$

Further, this can be numerically shown to go to 0 for any k that is not a multiple of N. I have written a function called `multiples(N)` that returns the multiples and non-multiple integers of some integer N. I then compute the RHS of Equation 2 for every non-multiple integer k of N, and write each solution into the text file `non_multiples_k.txt`. For $N = 100$, we can see that Equation 2 $\to 0$ for all such non-multiple integers k (as each is on the order of ~1e-15).

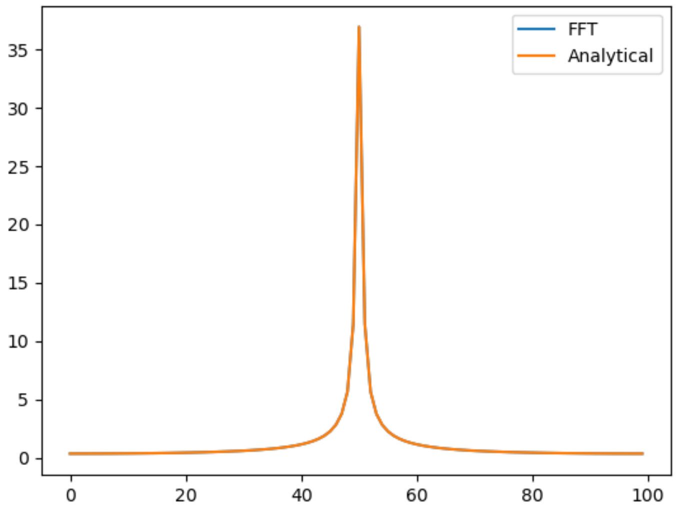

Figure 5: FFT of a sin function with non-integer k, using both NumPy's FFT and the analytical DFT of Equation 11. Spectral leakage is visible on either side of the peak.

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

$$F(k) = \sum_{x=0}^{N-1} f(x)e^{-2\pi ik'x/N} = \sum_{x=0}^{N-1} \sin(2\pi kx/N)e^{-2\pi ik'x/N}, \tag{6}$$

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

$$sin(2\pi kx/N) = \frac{1}{2i}\left(e^{2\pi ikx/N} - e^{-2\pi ikx/N}\right). \tag{7}$$

We can replace this form into Equation 6:

$$F(k) = \sum_{x=0}^{N-1} \frac{1}{2i}\left(e^{2\pi ikx/N} - e^{-2\pi ikx/N}\right) e^{-2\pi ik'x/N}, \tag{8}$$

which allows us to break up the sum as

$$F(k) = \frac{1}{2i}\left(\sum_{x=0}^{N-1} e^{2\pi ikx/N} e^{-2\pi ik'x/N} - \sum_{x=0}^{N-1} e^{-2\pi ikx/N} e^{-2\pi ik'x/N}\right). \tag{9}$$

This further reduces to

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

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

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

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 in Figure 5 the FFT and analytical DFT from Equation 11 of $sin(2\pi k/N)$ for $k = 50.125$. Spectral leakage is clearly visible, as the FFT and DFT are not delta functions with a peak at k, but there is some "spreading" of the peak on either side. The standard deviation between the FFT and the analytical DFT is 1.12e-13, which is only a few orders of magnitude from machine precision (∼1e-16 for Python), and the average residual is 2.26e-14.

(d) Here we will multiply the window function `cos_wind(x)` $= 0.5 - 0.5\cos(2\pi x/N)$ by our original sin function $\sin(2\pi kx/N)$ (noting again that k in a non-integer) and look at its FFT to see if the window function has reduced the spectral leakage. The FFT's of the "unwindowed" sin function and the "windowed" sin function are shown in Figure 6, where we can clearly see a sharper peak in the windowed FFT.

(e) To show that the FT of the window is $[N/2, -N/4, 0, 0, \ldots, 0, 0, -N/4]$, 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.

We can get the windowed FT by appropriately scaling the unwindowed FT using the information about the FT of the window. If we call our unwindowed FT $F(k)$, we can get the windowed FT, (let's call it $F'(k)$) simply by writing a for loop with the following structure, where we scale each point and subtract it from its scaled neighbors:

$$F'(k_i) = -\frac{1}{4}F(k_{i-1}) + \frac{1}{2}F(k_i) - \frac{1}{4}F(k_{i+1}) \tag{12}$$

In fact, this "neighbors" FT gets pretty close to the windowed FT; I have plotted the FT of the sin function with no window, after being multiplied by a window, and using the routine of Equation 12 in Figure 7. The peaks of the windowed FT and the "neighbors" FT are not exactly the same, but the "neighbors" FT also seems to get rid of the spectral leakage like the windowed FT.

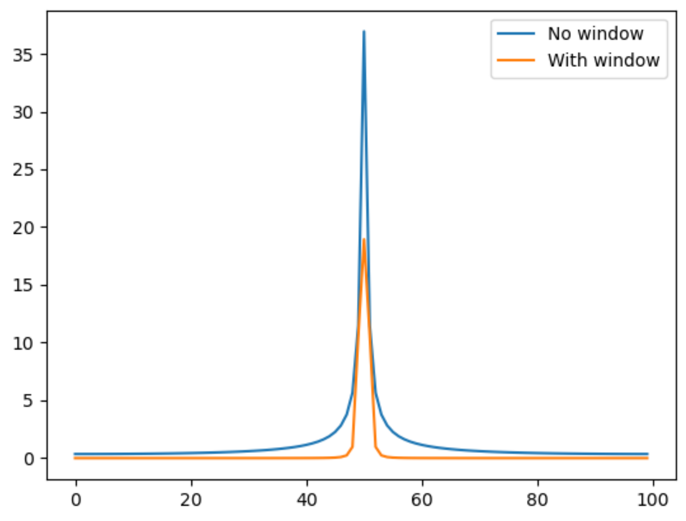

Figure 6: FFT of $\sin(2\pi kx/N)$ with and without a window function. We can see that the window function reduces spectral leakage.

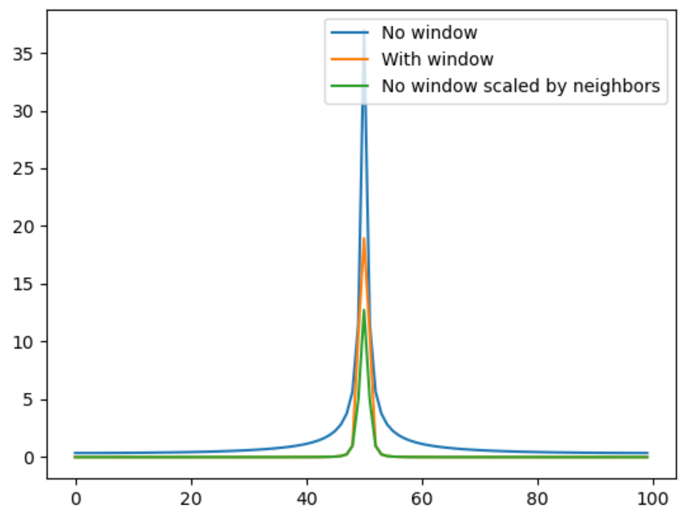

Figure 7: FFT of $\sin(2\pi kx/N)$ with and without a window function, as well as the "neighbors" FT using Equation 12. The "neighbors" FT also reduces the spectral leakage significantly.

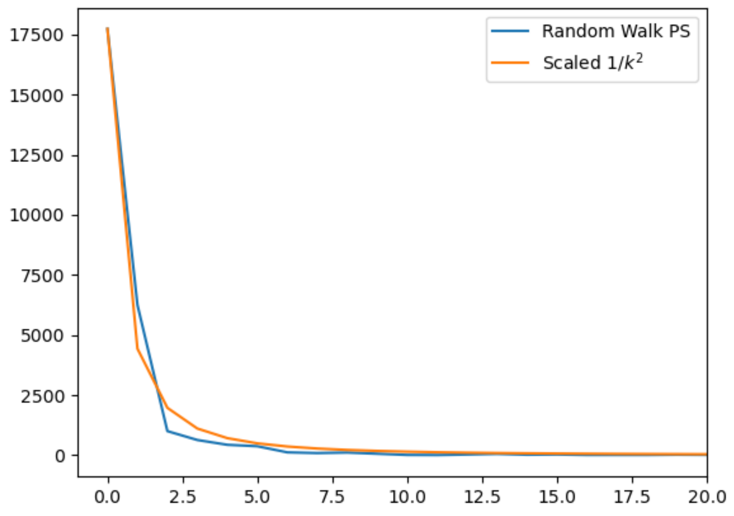

Figure 8: The power spectrum of a random walk compared with a scaled $1/k^2$. They appear to have similar behavior.

6. (a) If we have a random walk $f(x)$, the variance between one step and the next should be a Gaussian random number; we may then state

$$ndx = \left\langle (f(x) - f(x+dx))^2 \right\rangle . \tag{13}$$

Expanding this:

$$ndx = \langle f^2(x) - 2f(x)f(x+dx) + f^2(x+dx) \rangle \tag{14}$$

If we "walk" for long enough, we should find that the expectation values $f^2(x)$ and $f^2(x+dx)$ should be roughly equal; we will set these both equal to N and simplify:

$$\langle f(x)f(x+dx) \rangle = N - ndx/2 \tag{15}$$

This appears to be a correlation of two points separated by dx. At this point, it is my understanding that we must take the FT of $ndx/2$, but I don't really see how this can end up going like $\sim 1/k^2$; I believe dx is supposed to be some constant, but its FT is

$$F(k) = \sum_{x=0}^{N-1} \frac{ndx}{2} e^{-2\pi ikx/N}, \tag{16}$$

i.e. the FT of a constant, which is supposed to be a delta function. There must be something I'm missing with the correlation $\langle f(x)f(x+dx) \rangle$ above, which is supposed to be proportional to $c - |dx|$, where c is some large number. Does it make sense to take a FT of a correlation, as it seems we are implied to do? I think I'll have to wait for the solutions to come out to fully appreciate this!

(b) As suggested by the prompt, I created a random walk with `np.cumsum` and random numbers, the power spectrum of which is plotted in Figure 8 and saved to `rw_ps.png`. It appears that the power spectrum of the random walk does indeed go like $\sim 1/k^2$.