## #1.

Let's say we want to shift to the right the elements of an array $f(x)$ of size $N$ by some integer $n$ where $0 \leq n \leq N-1$. We can do this by invoking the shifting property of a DFT and the convolution theorem:

$$\begin{aligned} f(x-n) &= \mathcal{F}^{-1}\left\{\mathcal{F}(f(x-n))\right\} \\ &= \mathcal{F}^{-1}\{e^{\frac{-2\pi i n}{N}}\mathcal{F}(f(x))\} \\ &= \mathcal{F}^{-1}\{\mathcal{F}(\delta_n)\cdot\mathcal{F}(f(x))\} \\ &= \mathcal{F}^{-1}\{\mathcal{F}(\delta_n * f(x))\} \\ &= \delta_n * f(x) \end{aligned}$$

So, to shift our array, we compute its convolution with delta of $n$:

$$f(x-n) = \sum_{j=0}^{N-1} \delta_n(j) f(x-j)$$

As an example, take $f$ to be an ordered range from 0 to 199 and shift it to the right by 50:

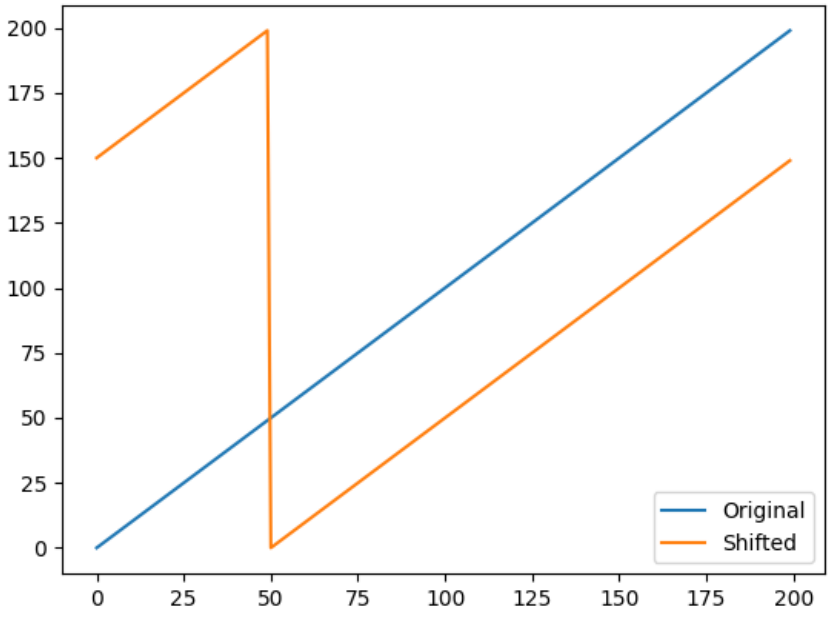


Figure 1: Shifting an array of ordered integers with size 200 by 50 elements to the right

## #2.

Computing the cross-correlation using

$$f \star g = \mathcal{F}^{-1}\{\mathcal{F}(f) \cdot \mathcal{F}^*(g)\}$$

is done quite straightforwardly using `Numpy`'s fft library and its complex number implementation. If we use this to cross-correlate a gaussian with itself, we see that we get yet another gaussian (Figure 2).

However, if we cross-correlate a gaussian with a shifted version of itself (for example, to the right by $n$), we again see the same gaussian as obtained above, but shifted to the *left* by $n$. This is easily seen when looking at our expression above:

$$\begin{aligned}
\mathcal{F}(f \star g) &= \mathcal{F}(f(x)) \cdot \mathcal{F}^*(f(x-n)) \\
&= \mathcal{F}(f(x)) \cdot \left(e^{\frac{-2\pi i n}{N}} \mathcal{F}(f(x))\right)^* \\
&= e^{\frac{2\pi i n}{N}} \cdot \mathcal{F}(f(x)) \cdot \mathcal{F}^*(f(x)) \\
&= e^{\frac{2\pi i n}{N}} \mathcal{F}(f \star f)
\end{aligned}$$

And so, because of the shifting propery, $f \star g$ is simply the shifted version of $f \star f$. The fact that it was shifted to the left comes from taking the complex conjugate, which gets rid of the minus sign in the complex exponential, hence the leftward shift. Note that in Figure 2, I shifted everything by 100 to the right so that the first gaussian is centred.

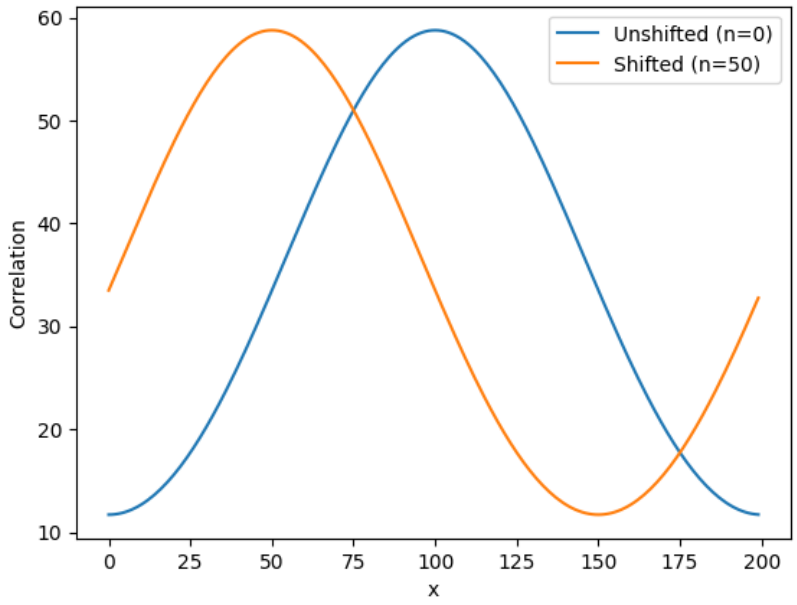


Figure 2: Cross-correlations between a gaussian and a shifted version of itself

## #3.

Because of the discrete nature of DFTs, the property that $F(-k) = F(k - N)$ which interferes with our result. Indeed, if our original domain goes from 0 to $N$, then we would "contaminate" half of our domain (from $N/2$ to $N$). To deal with this, we can double our domain and pad the second half with zeros. Since we are only interested in $k = 0$ to $N$, we can do our FFTs normally and then disregard the second half. We can test this on our FFT based convolution by convoluting a gaussian with itself:

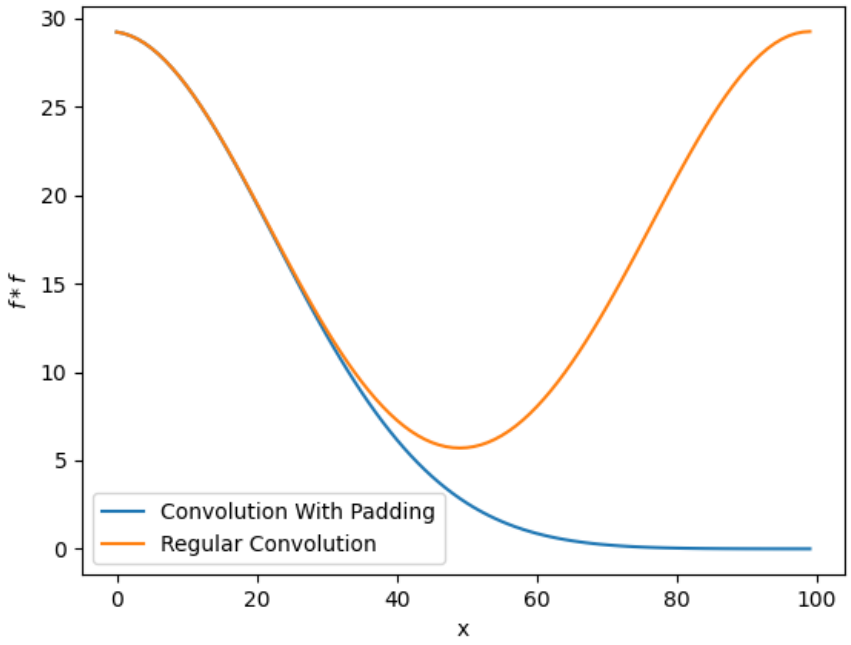


Figure 3: Convolution of gaussian with itself with and without padding

## #4.

We can re-write the sum as a geometric series:

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

If $k \neq 0$ or a multiple of $N$, then $r = |\exp(-2\pi ik/N)| < 1$ since $k \neq mN$ ($m \in \mathbb{Z}$) and thus this series converges to:

$$\frac{1-r^N}{1-r} = \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)}$$

If $k \in \mathbb{Z}$, then clearly $\exp(-2\pi ik) = 1$ while the denominator is non-zero. Hence, the sum converges to 0. By using l'Hôpital's rule, we can compute the limit as $k$ goes to zero. We find:

$$\lim_{k\to 0} \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)} = \lim_{k\to 0} \frac{2\pi i \cdot \exp(-2\pi ik)}{2\pi i/N \cdot \exp(-2\pi ik/N)} = \frac{2\pi i \cdot 1}{2\pi i/N \cdot 1} = N$$

Indeed, when $k = 0$ (or a multiple of $N$ for that matter), our sum is

$$\sum_{x=0}^{N-1} \exp(-2\pi ixk/N) = \sum_{x=0}^{N-1} 1 = N$$

We can use this to estimate analytically the DFT of of a sine wave with non-integer wave number $l$:

$$f(x) = \sin(2\pi ilx/N) = \frac{1}{2i}(e^{2\pi ilx/N} - e^{-2\pi ilx/N})$$

$$\Rightarrow F(k) = \frac{1}{2i}\sum_{x=0}^{N-1}(e^{2\pi ilx/N} - e^{-2\pi ilx/N})e^{-2\pi ikx/N}$$

$$= \frac{1}{2i}\sum_{x=0}^{N-1} e^{-2\pi ix(k-l)/N} - e^{-2\pi ix(k+l)/N}$$

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

As we should expect, we have poles at $k = \pm l$ Comparing this analytical FFT with `Numpy`'s, we find that they match pretty well, though not perfectly, certainly not to machine precision (Figure 4). While, we expect two delta functions ($k = l$ and $k = N - l$), we see that the peak is quite broad for a delta function!

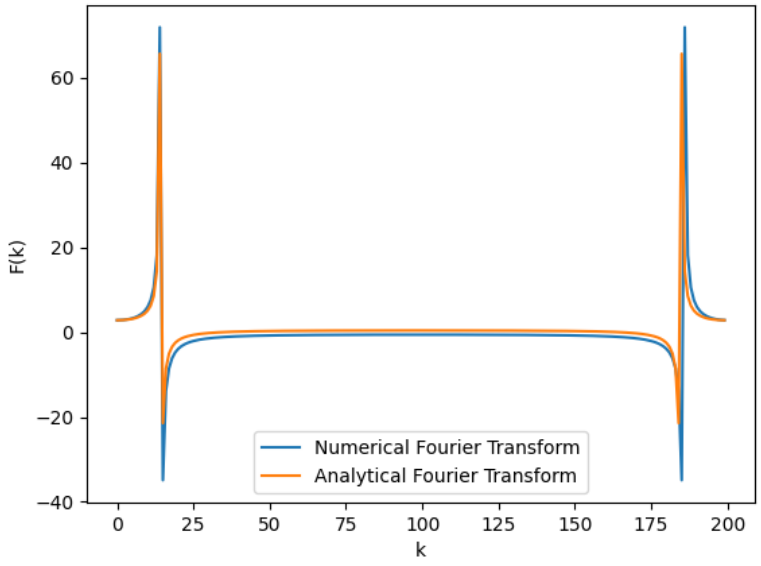


Figure 4: Comparison between numerical and analytical FFT of sine wave ($l = 14.33$)

If we now introduce a window function $w(x) = \frac{1}{2} - \frac{1}{2}\cos(2\pi x/N)$ which goes to zero at the edges and peaks at one in the middle, we can multiply our original sine wave with it in order to cancel the jump at the edge that is responsible for our spectral leakage. Indeed, by taking the DFT of our sine wave times the window function, we see that the peak is much narrower (Figure 5) than before, though its shape is different since $\mathcal{F}(f)$ is now convolved with $\mathcal{F}(w)$

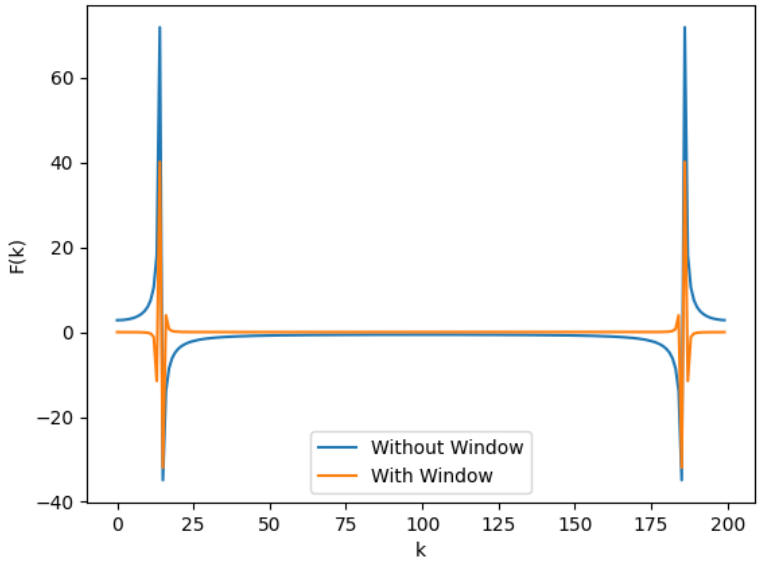


Figure 5: Comparison between DFTs of a sine wave with and without a window

Let's now compute the Fourier transform of the window function:

$$\begin{aligned}
\mathcal{F}(w) &= \sum_{x=0}^{N-1}\left(\frac{1}{2}-\frac{1}{2}\cos(2\pi x/N)\right)\exp(-2\pi ixk/N)\\
&= \sum_{x=0}^{N-1}\left(\frac{1}{2}-\frac{1}{4}\left(\exp(2\pi ix/N)+\exp(-2\pi ix/N)\right)\right)\exp(2\pi ixk/N)\\
&= \frac{1}{2}\sum_{x=0}^{N-1}\exp(-2\pi ixk/N)-\frac{1}{4}\sum_{x=0}^{N-1}\exp(-2\pi ix(k-1))-\frac{1}{4}\sum_{x=0}^{N-1}\exp(-2\pi ix(k+1))\\
&= \frac{1}{2}\cdot N\delta(0)-\frac{1}{4}\cdot N\delta(1)-\frac{1}{4}\cdot N\delta(-1)\\
&= \frac{N}{2}\delta(0)-\frac{N}{4}\delta(1)-\frac{N}{4}\delta(N-1)
\end{aligned}$$

So, the array representing the Fourier transform of $w$ would look like:

$$\left[\frac{N}{2},\frac{-N}{4},0,\cdots,0,\frac{-N}{4}\right]$$

This can be used to compute the windowed Fourier transform directly from the un-windowed one: by the Convolution Theorem, the Fourier transform of a product of functions is the convolution of their Fourier transform divided by $N$ (without this factor, you'd instead $\mathcal{F}^{-1}\{\mathcal{F}(f\cdot g)\}=Nf\cdot g$, which is not correct). Hence, we need to convolve $\mathcal{F}(w)$ with $\mathcal{F}(f)$ and divide by $N$:

$$\begin{aligned}
\mathcal{F}(f\cdot w) &= \frac{1}{N}\mathcal{F}(w)\star\mathcal{F}(f)=\frac{1}{N}W(k)\star F(k)\\
&= \frac{1}{N}\sum_{j=0}^{N-1}W(j)\cdot F(k-j)\\
&= \frac{1}{2}F(k)-\frac{1}{4}F(k-1)-\frac{1}{4}F(k-1+N)
\end{aligned}$$

So, we combine each entry in $F(k)$ with its immediate neighbours in the above manner to get the windowed Fourier transform. As seen in Figure 6, this agrees with the one computed using `Numpy`

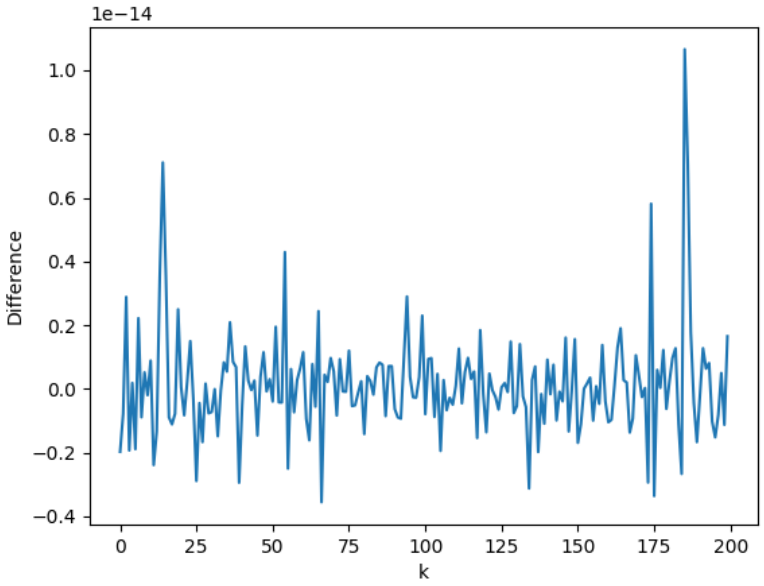


Figure 6: Difference between the numerical windowed FFT and the one obtained using unwindowed FFT