1.
```
def roll(array,n):
    N=len(array)
    k=np.arange(N)
    return np.real(ifft(fft(array)*np.exp(-2j*np.pi*n*k/N)))
```

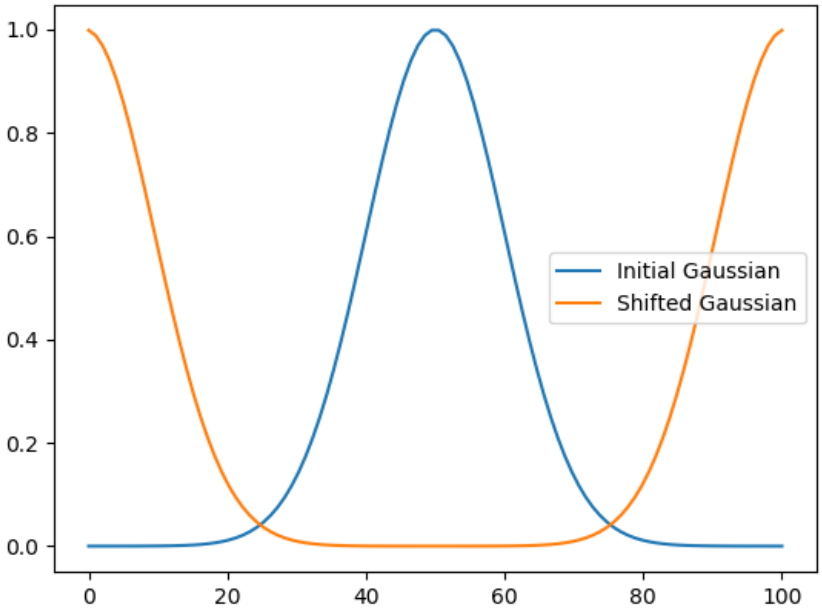

Figure 1: Centered Gaussian and Gaussian shifted by half the length of the array.

2. The function corr takes the correlation function of two arrays. In Figure 2, the correlation of a Gaussian with itself is shown.

```
def corr(f,g):
    return irfft(rfft(f)*np.conjugate(rfft(g)))
```

3. Using corr(y,roll(y,n)) gives the correlation of an array, y, with itself shifted by n. The results are shown in Figure 2 for a Gaussian and several values of n. The

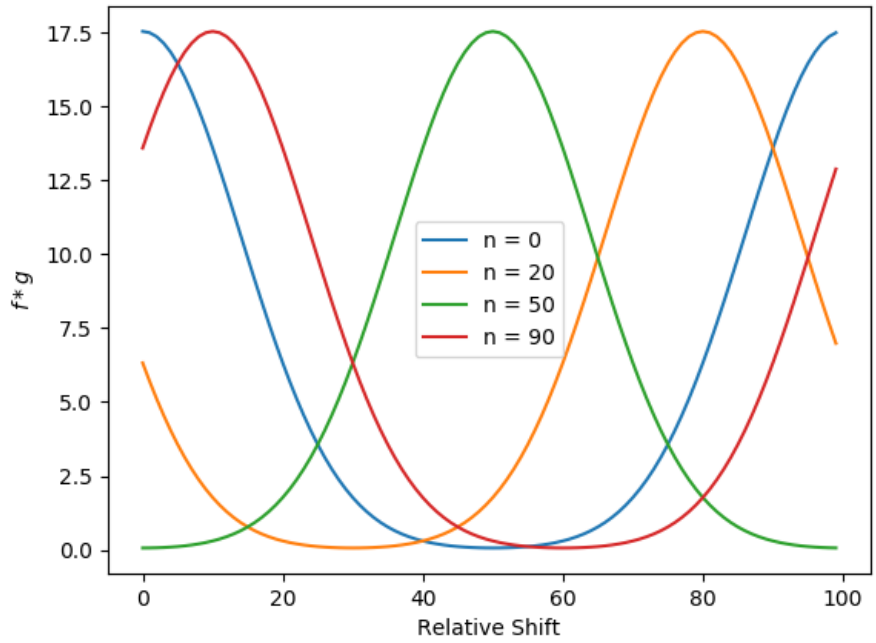

Figure 2: Correlation of a Gaussian with a shifted Gaussian. The n=0 case corresponds to the correlation of the function with itself, and resembles the result of Figure 1.

4. The function corr_pad pads the arrays with zeros and returns the correlation function with the shift running from zero (functions on top of each other) to the length of the array. It could easily be modified to run from -length to length by padding both sides.

```
def corr_pad(f,g):
    N=min(len(f),len(g))
    f_pad, g_pad=np.zeros(2*len(f)),np.zeros(2*len(g))
    f_pad[:len(f)],g_pad[:len(g)]=f,g
    return irfft(rfft(f_pad)*np.conjugate(rfft(g_pad)))[:N]
```

5. (a)

$$\begin{aligned}
\sum_{x=0}^{N-1} \exp(-2\pi ikx/N) &= \sum_{x=0}^{N-1} (\exp(-2\pi ik/N))^x \\
&= \sum_{x=0}^{N-1} \alpha^x, \quad \text{where} \quad \alpha = \exp(-2\pi ik/N), \\
&= \frac{1-\alpha^N}{1-\alpha}, \quad \text{if} \quad |\alpha| < 1, \\
&= \frac{1-(\exp(-2\pi ik/N))^N}{1-\exp(-2\pi ik/N)}, \\
&= \frac{1-\exp(-2\pi ik)}{1-\exp(-2\pi ik/N)}.
\end{aligned}$$

(b) For integer k, $\exp(-2\pi ik) = 1$, so the numerator is 0. For non-zero denominator, the sum is zero. For $k \to 0$, the value is N.

$$\begin{aligned}
\lim_{k\to 0} \sum_{x=0}^{N-1} \exp(-2\pi ikx/N) &= \sum_{x=0}^{N-1} \exp(0) && (1) \\
&= \sum_{x=0}^{N-1} 1 && (2) \\
&= N && (3)
\end{aligned}$$

(c) Take the function $f(x) = 2\cos(2\pi k'x/N) = \exp(2\pi ik'x/N) + \exp(-2\pi ik'x/N)$.

$$\begin{aligned}
DFT\{f(x)\} &= \sum_{x=0}^{N-1} f(x) \exp(-2\pi ikx/N), \quad \text{for N points and} \quad 0 < k < N \\
&= \sum_{x=0}^{N-1} (\exp\big(2\pi ik'x/N\big) + \exp\big(-2\pi ik'x/N\big)) \exp(-2\pi ikx/N) \\
&= \sum_{x=0}^{N-1} \exp\big(-2\pi i(kx + k'x)/N\big) + \sum_{x=0}^{N-1} \exp\big(-2\pi i(kx - k'x)/N\big) \\
&= \frac{1-\exp(-2\pi i(k+k'))}{1-\exp(-2\pi i(k+k')/N)} + \frac{1-\exp(-2\pi i(k-k'))}{1-\exp(-2\pi i(k-k')/N)}.
\end{aligned}$$

The DFT has value N when $(k \pm k')$=0, and 0 otherwise. Figure 3 shows the result for integer and non integer values.

```
T=1. #acquisition time
F=100. #sampling frequency
N=T*F
```

```
t=np.arange(0,T,1/F)
f=np.arange(-F/2, F/2, 1/T)
mycos = lambda cycles: 2*np.cos(2*np.pi*(cycles*F)*t/N)
window = 0.5-0.5*np.cos(2*np.pi*t*F/N)
#plt.plot(t,window)
plt.plot(f,np.abs(fft(mycos(4.5))), label="4.5 cycles")
plt.plot(f,np.abs(fft(mycos(9))), label="9 cycles")
plt.plot(f,np.abs(fft(mycos(4.5)*window)), label="4.5 cycles windowed")
```

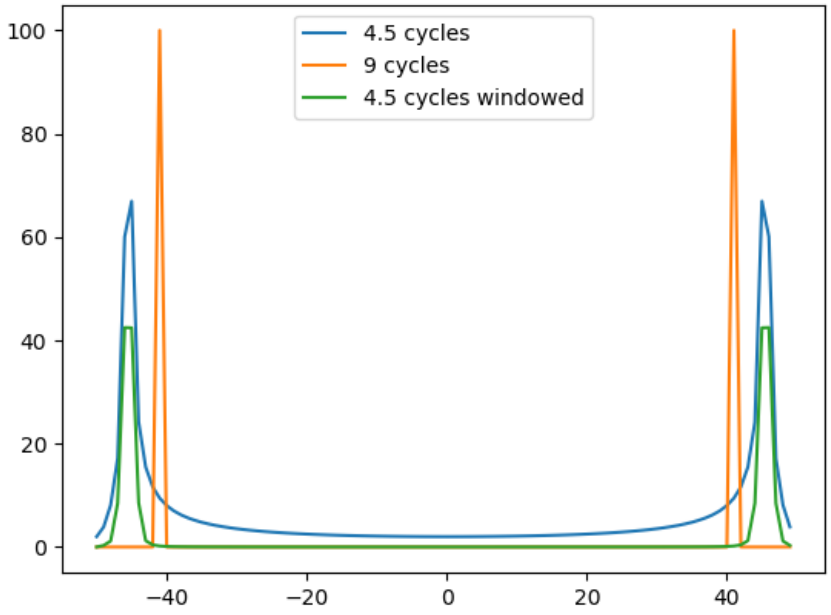

Figure 3: For an integer number of cycles of the sinusoid, the dft reaches the full height of N. For a non-integer number, the peaks are broadened to nearby frequencies. Windowing reduces the broadening, but also the amplitude (though I may have a weird factor of 2 somewhere).

(d) See Figure 3.

(e) See Figure 4.

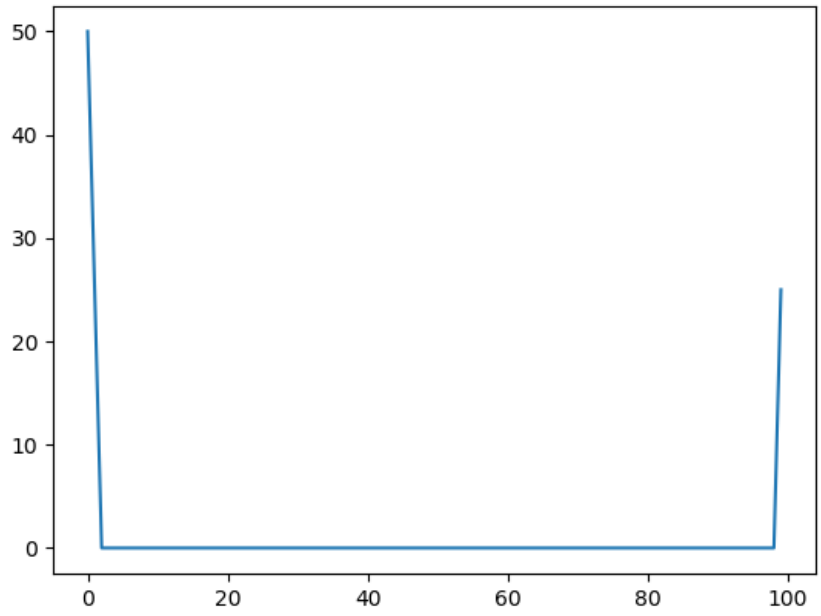

Figure 4: FFT of the window used above. Needs some sign checks! Could be replaced by averaging endpoints in Fourier space.