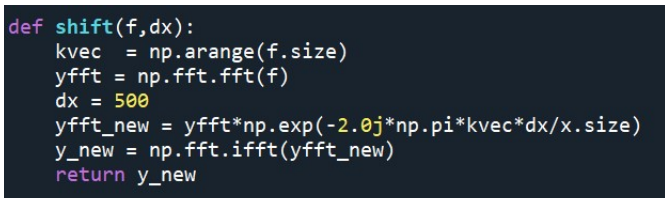
Here is the report of PHYS512 problemset 4 written by Jiaxing MA.

problem1

Here is the function to shift the function, the argument f is the function, and dx is the shift length.



```
def shift(f,dx):
    kvec  = np.arange(f.size)
    yfft = np.fft.fft(f)
    dx = 500
    yfft_new = yfft*np.exp(-2.0j*np.pi*kvec*dx/x.size)
    y_new = np.fft.ifft(yfft_new)
    return y_new
```

Figure 1

And here is the Gaussian function that shift by half of the range

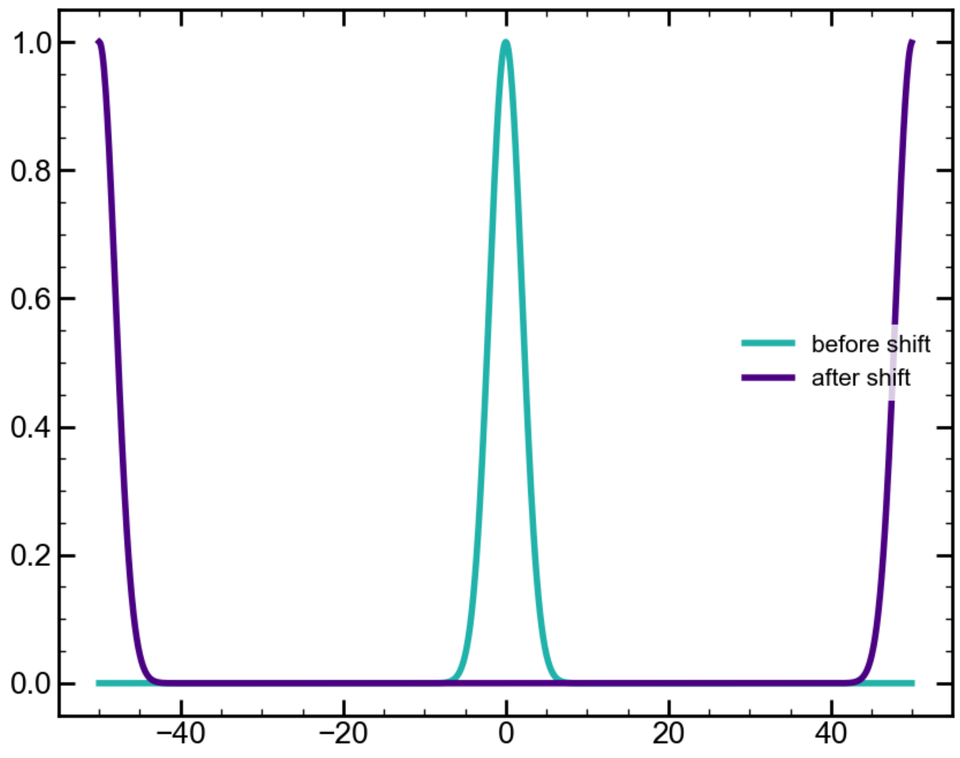
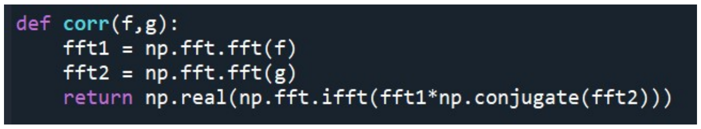


Figure 2

problem2

Here is the function that make correlation of two arrays.



```
def corr(f,g):
    fft1 = np.fft.fft(f)
    fft2 = np.fft.fft(g)
    return np.real(np.fft.ifft(fft1*np.conjugate(fft2)))
```

Figure 3

Here is the plot of Gaussian function take correlation by it self.

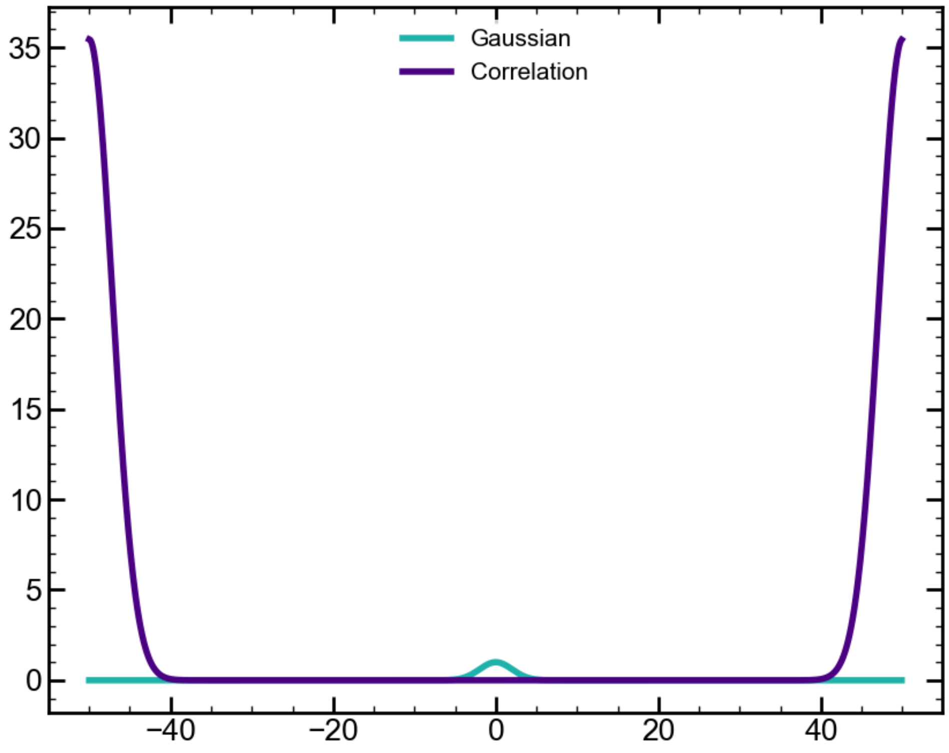


Figure 4

problem3

I combine the code from the 2 problem before, and make the correlation of a Gaussian and shifted Gaussian, here is the plot I got.

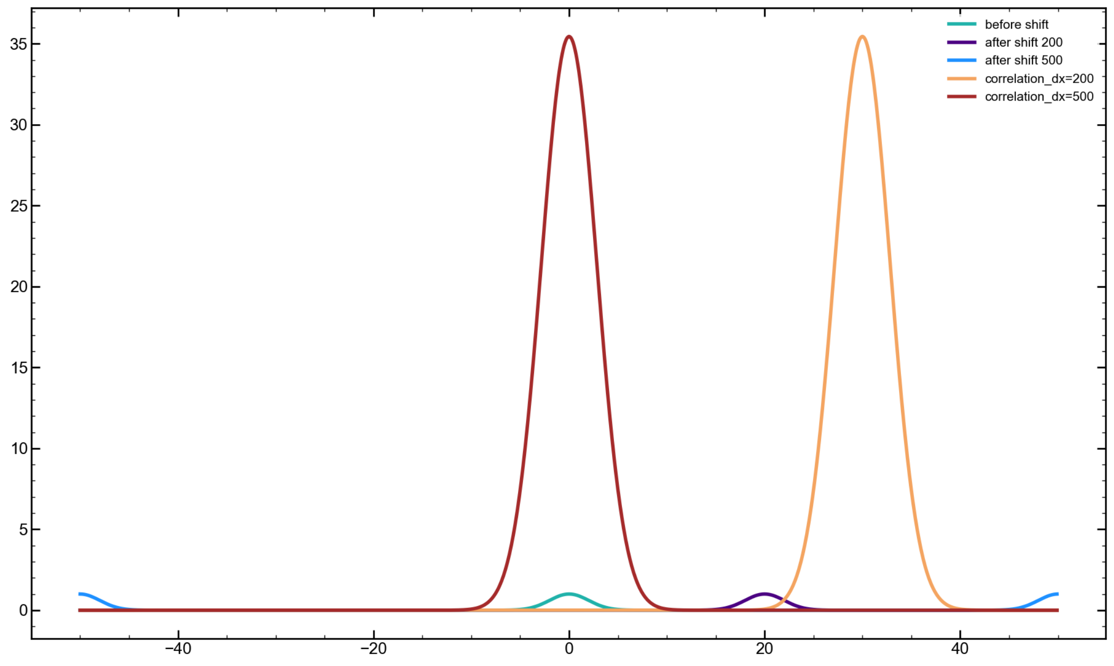
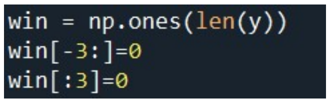


Figure 5

The correlation shifts when the shift distance of the other Gaussian changes. This makes sense, because the FFT of Gaussian is also a Gaussian, so when the Gaussian shift, the correlation will also shift.

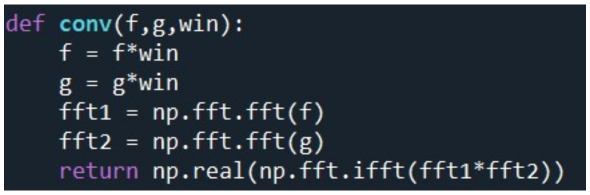
problem4

To avoid wrapping around, I wrote a routine that add a square window to the function, which is a array is 0 at both end and 1 for the other. Here is the code:



```
win = np.ones(len(y))
win[-3:]=0
win[:3]=0
```

Figure 6



```
def conv(f,g,win):
    f = f*win
    g = g*win
    fft1 = np.fft.fft(f)
    fft2 = np.fft.fft(g)
    return np.real(np.fft.ifft(fft1*fft2))
```

Figure 7

And here is the plot of convolution of two sine wave with different frequency with and without the square window.

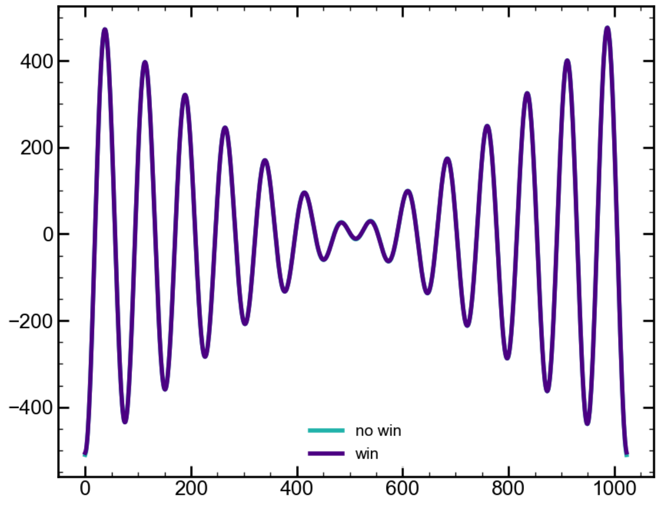


Figure 8: convolution of two sine wave with different frequency

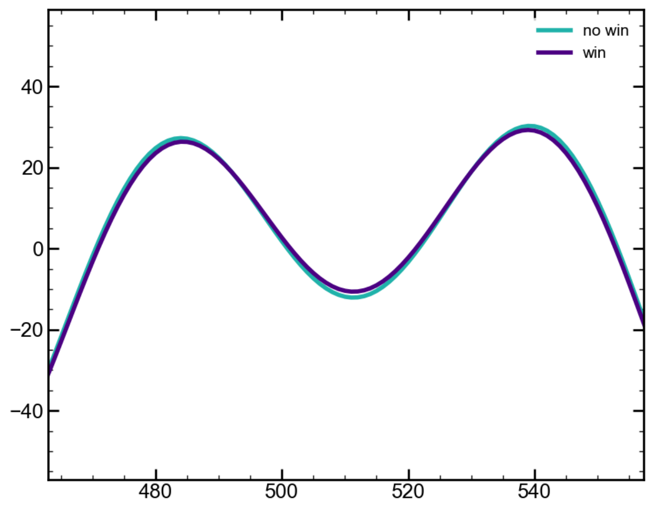


Figure 9: convolution of two sine wave with different frequency, zoom on curtain frequency.

As we can see, with the window, there is less wrap-around.

problem5

a

We can use α to represent $e^{-2\pi ikx/N}$, the left of the equation can be written as:

$$\begin{aligned}\sum_{x=0}^{N-1} e^{-2\pi ikx/N} &= \sum_{x=0}^{N-1}\alpha^x \\ &= 1+\alpha+\alpha^2+.......+\alpha^{N-1} \\ &= \frac{1-\alpha^N}{1-\alpha} \\ &= \frac{1-e^{-2\pi ik}}{1-e^{-2\pi ikx/N}} \\ &= right\end{aligned}$$

b

When k approaches zero, we can use L'Hôpital's rule to find the approach of the equation.

$$\lim_{k\to 0}\frac{1-e^{-2\pi ik}}{1-e^{-2\pi ikx/N}} = \lim_{k\to 0}\frac{2\pi ie^{2\pi ik}}{2\pi ike^{-2\pi ikx/N}/N} = N$$

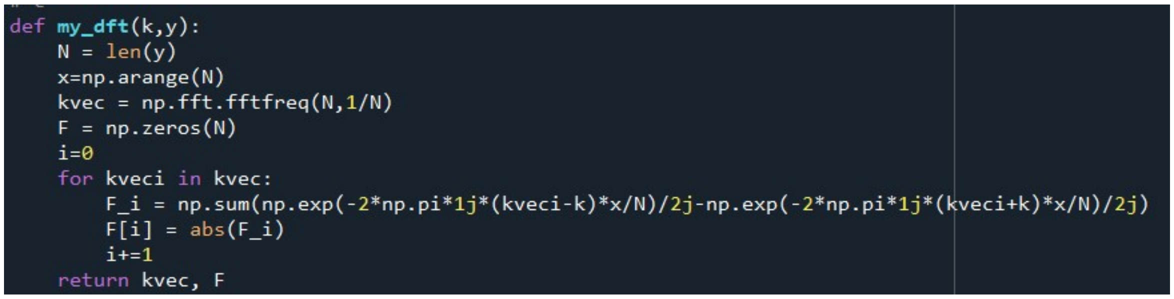
When k is a integer and is not a multiple of N, we have $e^{-2\pi ik}=1$ and $e^{-2\pi ik/N}\neq 1$, so for the equation, the denominator is not zero, and nominator is zero, so $\frac{1-e^{-2\pi ik}}{1-e^{-2\pi ikx/N}}=0$

c

To write down the DFT analytically, I use the similar sum as before,with a sine wave at frequency $2\pi k$ with a N points array.

$$\begin{aligned}DFT(k') &= \sum_{x=0}^{N-1} sin(2\pi ikx/N)e^{-2\pi ik'x/N} \\ &= \sum_{x=0}^{N-1}\frac{e^{2\pi ikx/N}-e^{2\pi ikx/N}}{2i}e^{-2\pi ik'x/N} \\ &= \sum_{x=0}^{N-1}(\frac{e^{-2\pi i(k'-k)x/N}}{2i}-\frac{e^{-2\pi i(k'+k)x/N}}{2i})\end{aligned}$$

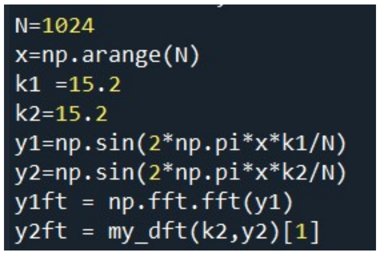
As we can see, the dft will be $N/2$ when $k'=k$ or $k'=-k$, and the other DFT will be 0 at other k'. Here is the DFT code I wrote.



```python
def my_dft(k,y):
    N = len(y)
    x=np.arange(N)
    kvec = np.fft.fftfreq(N,1/N)
    F = np.zeros(N)
    i=0
    for kveci in kvec:
        F_i = np.sum(np.exp(-2*np.pi*1j*(kveci-k)*x/N)/2j-np.exp(-2*np.pi*1j*(kveci+k)*x/N)/2j)
        F[i] = abs(F_i)
        i+=1
    return kvec, F
```

Figure 10

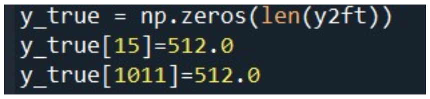
The function I use is a sine wave with non integer k, and I calculated the DFT use written DFT and numpy FFT.



```python
N=1024
x=np.arange(N)
k1 =15.2
k2=15.2
y1=np.sin(2*np.pi*x*k1/N)
y2=np.sin(2*np.pi*x*k2/N)
y1ft = np.fft.fft(y1)
y2ft = my_dft(k2,y2)[1]
```

Figure 11

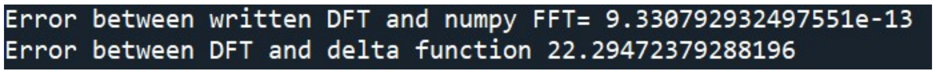
Here is the error I got compare to the FFT function from numpy, the error is $9.3e^{-}13$, and then I compare the result to the theoretical FFT of the sin wave, which is two delta function at frequency, and negative frequency of sine wave, here is the code of delta function.



```python
y_true = np.zeros(len(y2ft))
y_true[15]=512.0
y_true[1011]=512.0
```

Figure 12

And the error between the DFT and real fft is much large because of spectral leak, the error is 22.3. Here is the result of the code and the plot.



```
Error between written DFT and numpy FFT= 9.330792932497551e-13
Error between DFT and delta function 22.29472379288196
```

Figure 13

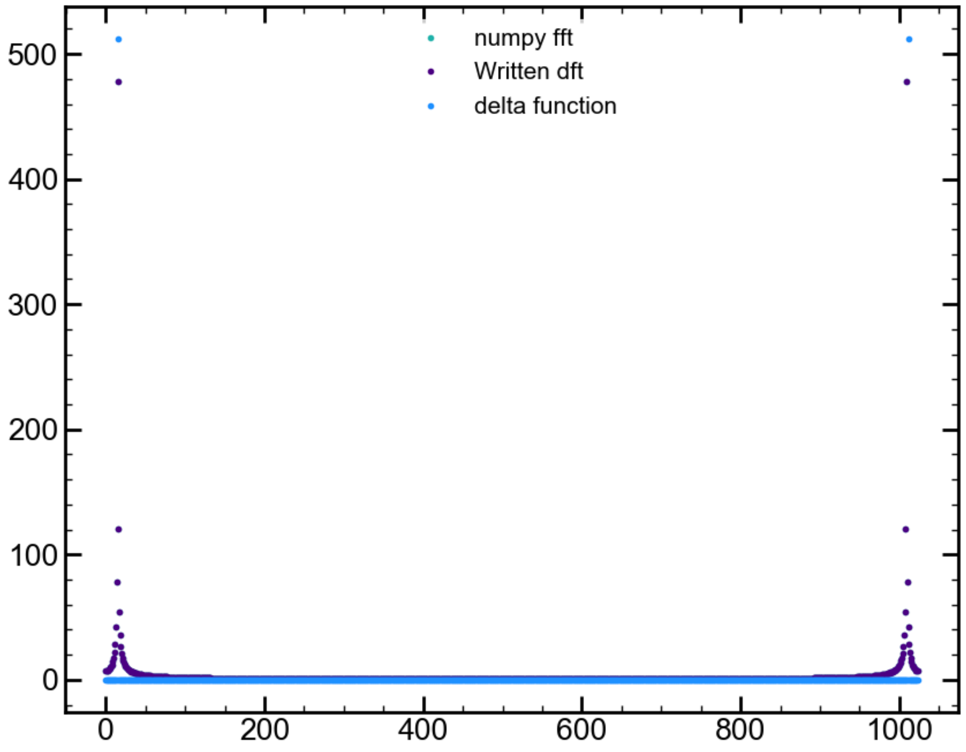
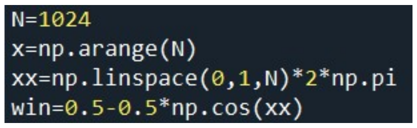


Figure 14

d

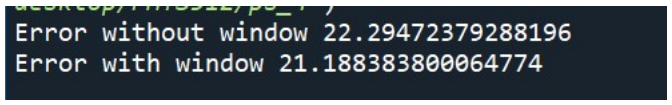
Here is the window function I used:



```
N=1024
x=np.arange(N)
xx=np.linspace(0,1,N)*2*np.pi
win=0.5-0.5*np.cos(xx)
```

Figure 15

And then I compare the error of DFT using numpy with the delta function with and without window, the error went down, and the spectral leak become smaller. Here is the result and plot.



```
Error without window 22.29472379288196
Error with window 21.188383800064774
```

Figure 16

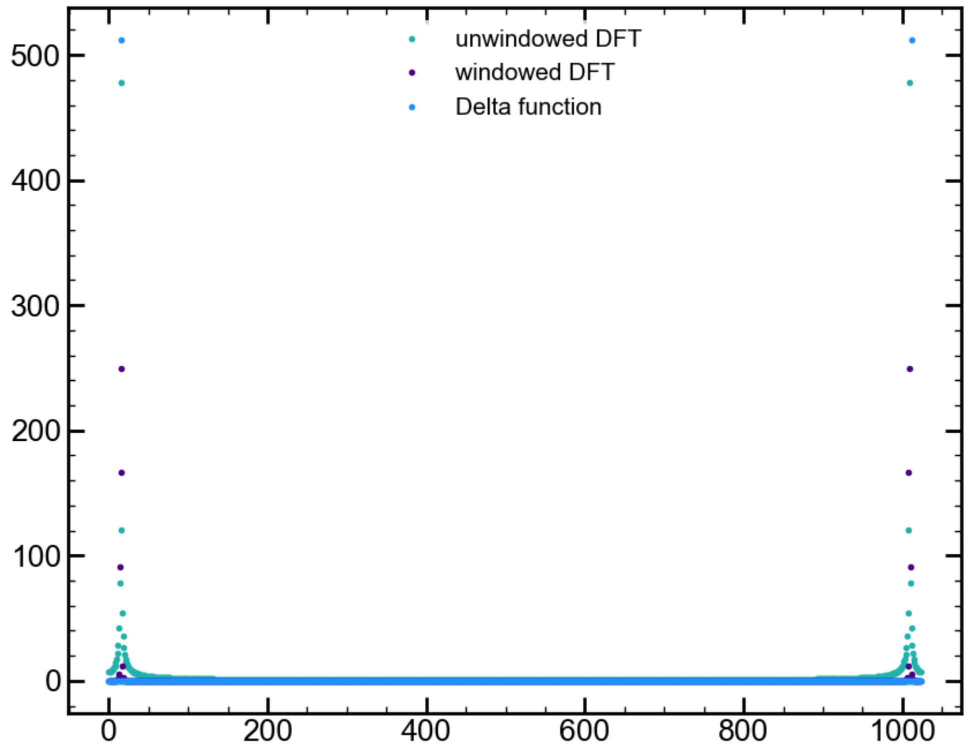


Figure 17

As we can see from the purple and green line, the spectral leak decrease a lot when I applied the window.

e

I calculated the FFT of window function theoretically.

$$\begin{aligned} FFT(k) &= \sum_{x=0}^{N-1}(0.5 - 0.5cos(2\pi ix/N))e^{-2\pi ikx/N} \\ &= \sum_{x=0}^{N-1}(0.5 - 0.5(\frac{e^{i2\pi x/N} + e^{-i2\pi x/N}}{2})e^{-2\pi ikx/N}) \\ &= \sum_{x=0}^{N-1}(0.5e^{-2\pi ikx/N} - 0.5e^{-2\pi i(k-1)x/N}/2 - 0.5e^{-2\pi i(k+1)x/N}/2) \end{aligned}$$

when $k \rightarrow 0$, $FFT(k) = 0.5N$; when $k \rightarrow 1$, $FFT(k) = -0.25N$; when $k \rightarrow -1$, $FFT(k) = -0.25N$, the rest of the FFT(k) is 0. So the FFT of window function is $[N/2, -N/4, 0, 0, 0, .., -N/4]$