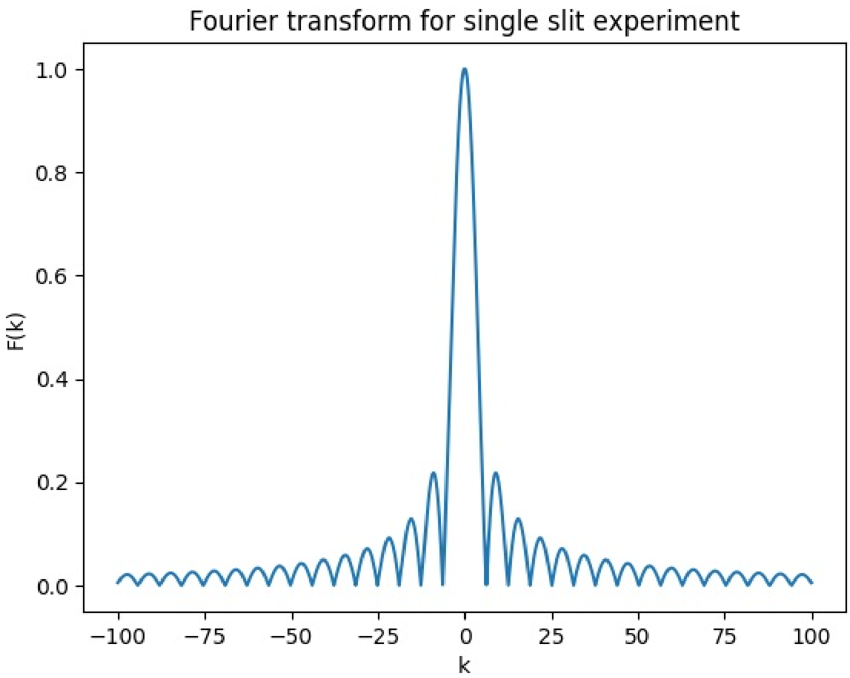
In [1]:
```
import numpy as np
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt
import os
from scipy.signal import find_peaks
from scipy import stats
from scipy import optimize
from scipy.optimize import curve_fit
from IPython.display import Image
from IPython.core.display import HTML
from scipy import signal
from scipy.signal import find_peaks
```

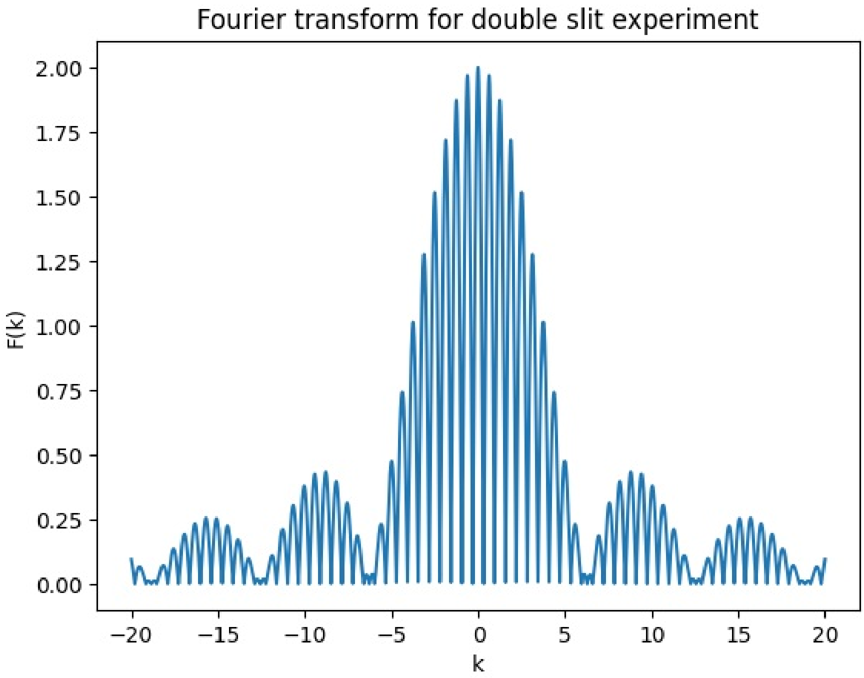
Nicolò De Masi 3746606

Task 0

In [192…
```
b=1
k=np.linspace(-100,100,10000)
f_k=2/k*np.sin(k*b/2)

plt.plot(k,np.abs(f_k))
plt.xlabel("k")
plt.ylabel("F(k)")
plt.title("Fourier transform for single slit experiment")
plt.show()
```



In [201…
```
b=1
g=10
k=np.linspace(-20,20,10000)
f_k=4/k*np.sin(k*b/2)*np.cos(k*g/2)

plt.plot(k,np.abs(f_k))
plt.xlabel("k")
plt.ylabel("F(k)")
plt.title("Fourier transform for double slit experiment")
plt.show()
```

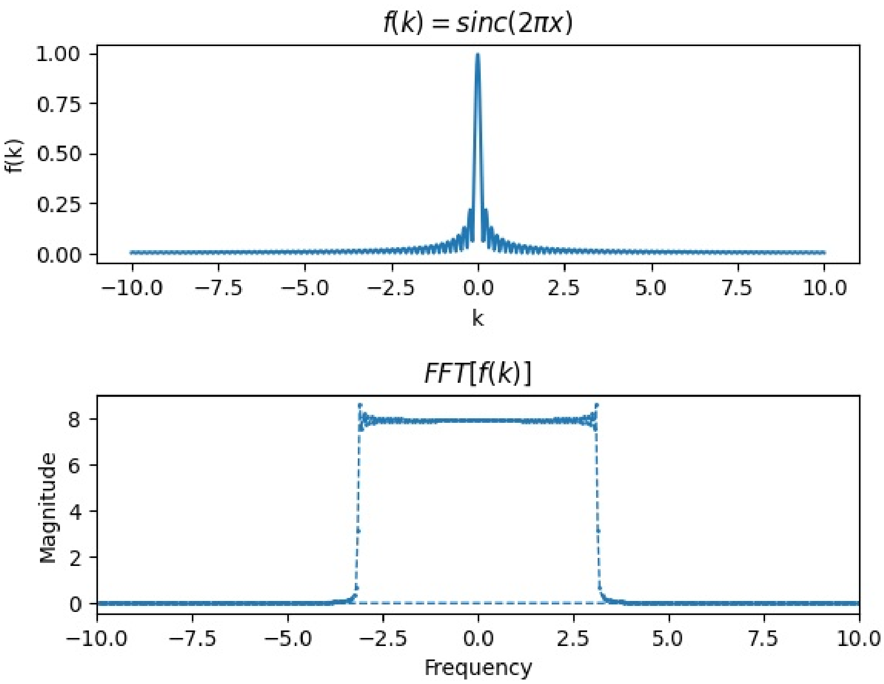


In [284…

```
# Define the x range
x = np.linspace(-10, 10, 1000)

# Calculate the y values for the sinc function
y = np.sinc(2*np.pi*x)

# Calculate the FFT of y
fft_y = np.fft.fft(y)


# Calculate the frequency axis
freq = np.fft.fftfreq(len(x), d=x[1]-x[0])


fig, (ax1, ax2) = plt.subplots(2)
fig.subplots_adjust(hspace=0.6)
ax1.plot(x, np.abs(y))
ax2.plot(freq, np.abs(fft_y), 'o--',lw=1, ms=1)
ax2.set_xlim([-10,10])

ax1.set_title('$f(k) = sinc(2\u03C0x)$')
ax2.set_title('$FFT[f(k)]$')
ax1.set(xlabel='k', ylabel='f(k)')
ax2.set(xlabel='Frequency', ylabel='Magnitude')
```

Out[284]:

```
[Text(0.5, 0, 'Frequency'), Text(0, 0.5, 'Magnitude')]
```

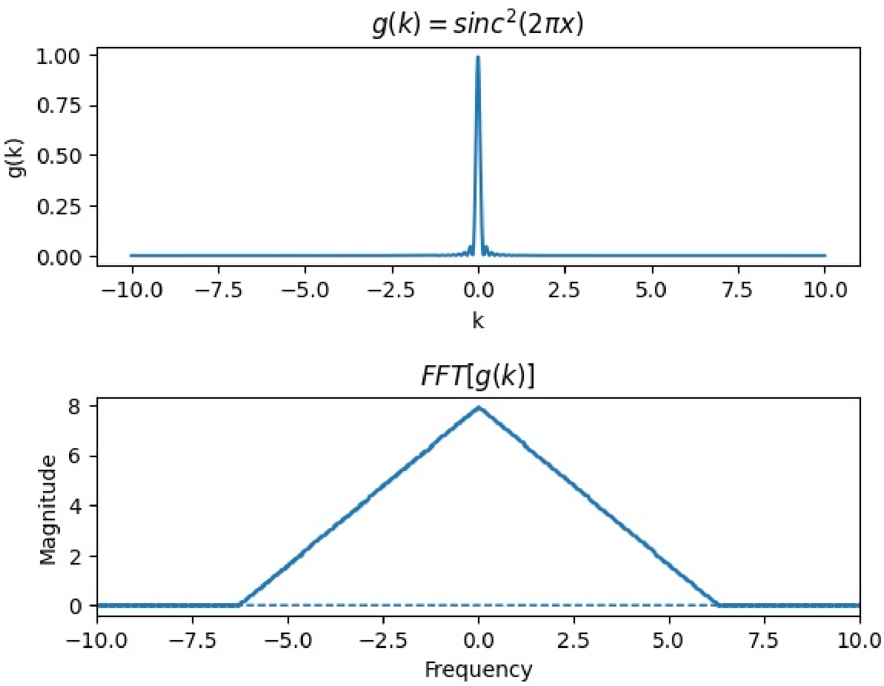


In [289…

```
#for square:
```

```
y_2=y**2

fft_y_2 = np.fft.fft(y_2)

fig, (ax1, ax2) = plt.subplots(2)
fig.subplots_adjust(hspace=0.6)
ax1.plot(x, np.abs(y_2))
ax2.plot(freq, np.abs(fft_y_2), 'o--',lw=1, ms=1)
ax2.set_xlim([-10,10])

ax1.set_title('$g(k) = sinc^2(2\u03C0x)$')
ax2.set_title('$FFT[g(k)]$')
ax1.set(xlabel='k', ylabel='g(k)')
ax2.set(xlabel='Frequency', ylabel='Magnitude')
```

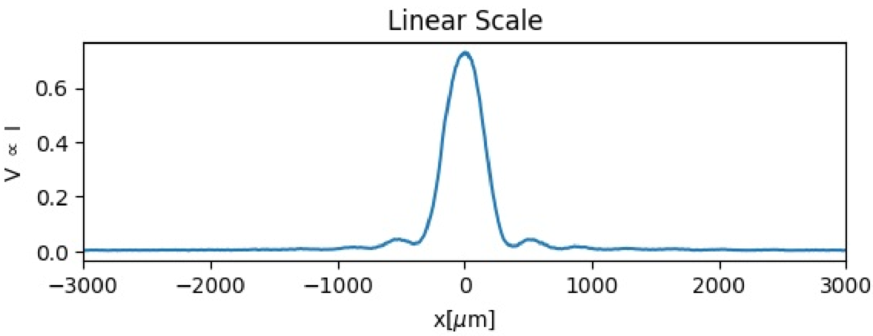
Out[289]: [Text(0.5, 0, 'Frequency'), Text(0, 0.5, 'Magnitude')]

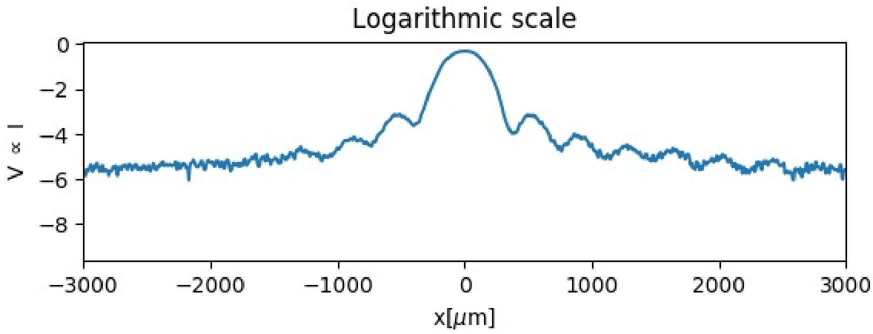


Task 1

In [3]:
```
np.seterr(divide='ignore')

df=pd.read_csv('1_lin.csv')
x=np.array(df['x'])
I=np.array(df['I'])

x=x*7
x = x - 7490

I_log=np.log(I)

fig, (ax1, ax2) = plt.subplots(2)
fig.subplots_adjust(hspace=0.6)

ax1.plot(x, I)
ax1.set(xlabel=r'x[$\mu$m]', ylabel=r'V $\propto$ I')
ax1.set_title('Linear Scale')
ax1.set_xlim([-3000,3000])

ax2.plot(x,I_log)
ax2.set_title('Logarithmic scale')
ax2.set(xlabel=r'x[$\mu$m]', ylabel=r'V $\propto$ I')
ax2.set_xlim([-3000,3000])
```

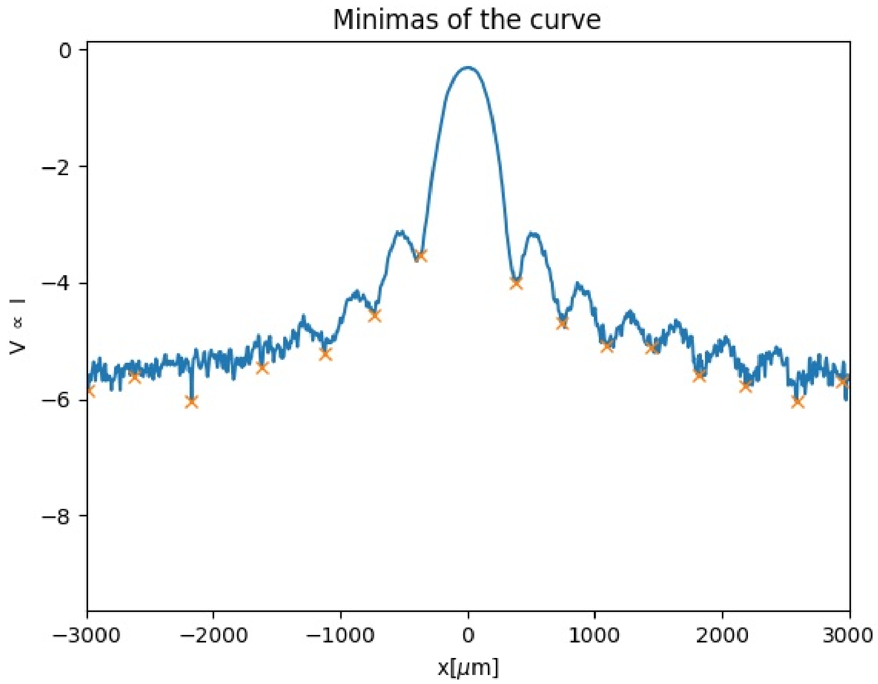
Out[3]: (-3000.0, 3000.0)





In [5]:

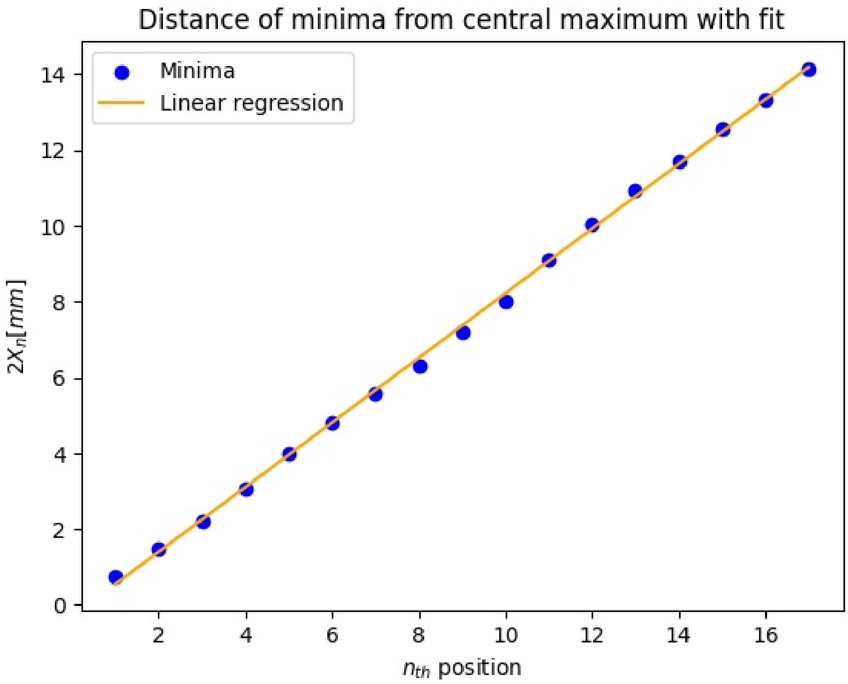
```
peaks, _ = find_peaks(-I_log, height=2, distance=50)

plt.plot(x,I_log)
plt.title("Minimas of the curve")
plt.plot(x[peaks], I_log[peaks], "x")
plt.xlabel(r'x[$\mu$m]')
plt.ylabel(r'V $\propto$ I')
plt.xlim([-3000,3000])
plt.show()

#we save the first 7 peaks in nth directions

I_peaks=I_log[peaks]

x_peaks=x[peaks]

p1=x_peaks[:17]

p2=x_peaks[17:34]

p1=p1[::-1]

#this is the lenghts Xn2

p=np.abs(p1)+np.abs(p2)

p=p*0.001

print(p)
```




```
[ 0.756  1.47   2.212  3.059  3.983  4.802  5.572  6.321  7.203  8.022
  9.093 10.038 10.92  11.725 12.551 13.335 14.154]
```

In [19]:

```
#set wavelenght:

l= 0.000636 #[mm]

#set focal distance:

f=300.8 #[mm]

n= np.array(list(range(1, 18)))

res = stats.linregress(n, p)
plt.scatter(n,p, color='blue', label= "Minima")
plt.plot(n, res.intercept + res.slope*n, label='Linear regression', color="orange")
plt.xlabel("$n_{th}$ position")
plt.ylabel("$2X_n [mm]$")
plt.title("Distance of minima from central maximum with fit")
plt.legend()
plt.show()

slope=res[0]

#from our formula, slope= 2*f*l/b -> b=2*f*l/slope

b=2*f*l/slope

print(b)
```

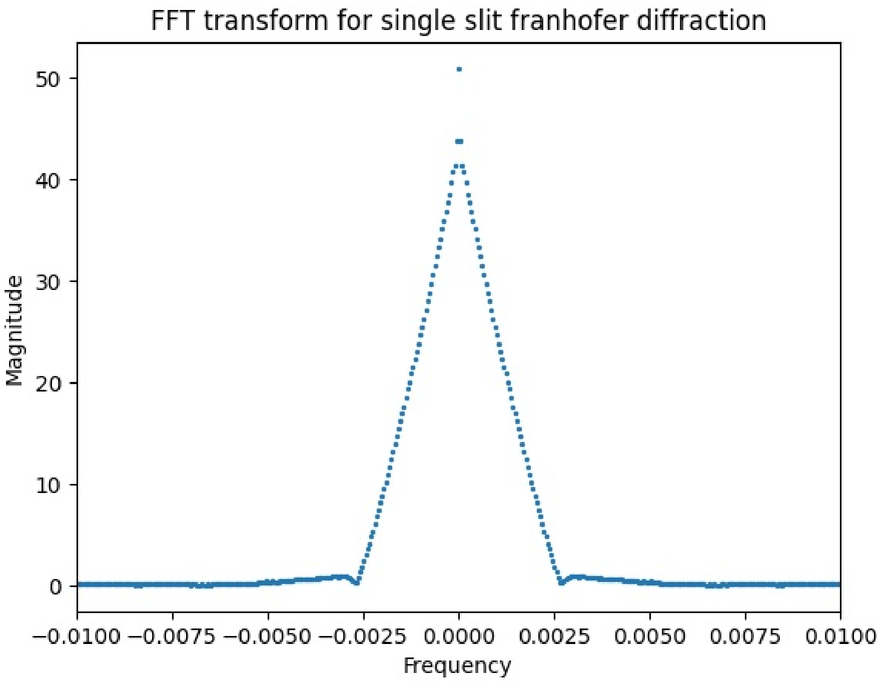



```
0.4488415270757501
```

In [81]:

```
fft_I = np.fft.fft(I)
freq = np.fft.fftfreq(len(x), d=x[1]-x[0])

peaks, _ = find_peaks(-fft_I)
first_minimum = np.min(np.abs(peaks))
slit_width = first_minimum * np.pi/ (x[1] - x[0])


print(slit_width)


plt.scatter(freq,np.abs(fft_I), s=2)
plt.xlim([-0.01,0.01])
plt.xlabel('Frequency')
plt.ylabel('Magnitude')
plt.title("FFT transform for single slit franhofer diffraction")
plt.show()
```

```
0.4487989505128276
```

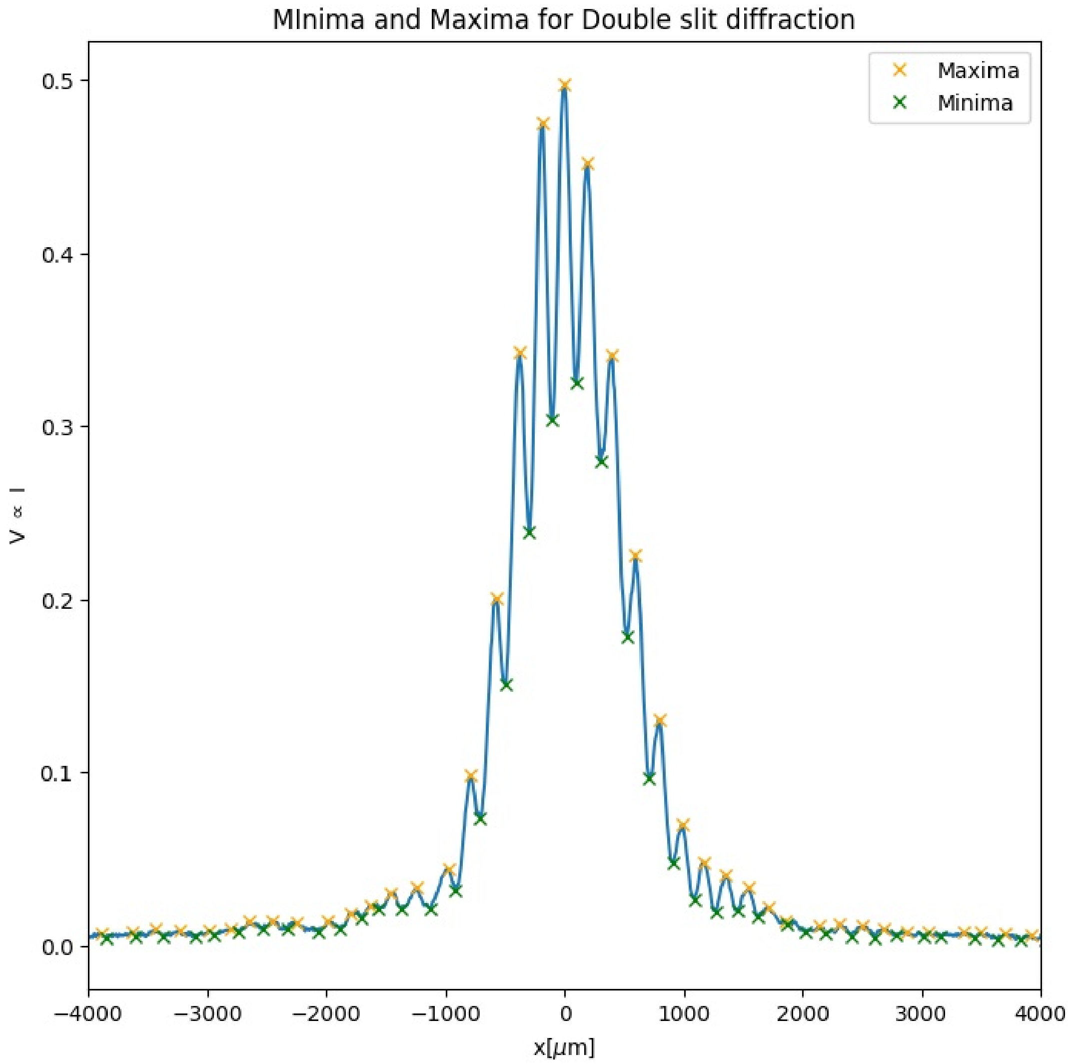


Task 2

In [153…

```
df=pd.read_csv('2_lin.csv')
x=np.array(df['x2'])
I=np.array(df['I2'])
```

```python
x=x*7
x=x-(7*1149)


maxima, _ = find_peaks(I, distance=20)
minima,_ = find_peaks(-I, distance=20)

plt.figure(figsize=(8, 8))
plt.plot(x,I)
plt.plot(x[maxima], I[maxima], "x", color = "orange", label="Maxima")
plt.plot(x[minima], I[minima], "x", color = "green", label="Minima")
plt.title("MInima and Maxima for Double slit diffraction")
plt.xlabel(r'x[$\mu$m]')
plt.ylabel(r'V $\propto$ I')
plt.legend()
plt.xlim([-4000,4000])
plt.show()

x_min=x[minima]
x_max=x[maxima]

p1m=x_min[:40]

p2m=x_min[40:80]

p1M=x_max[:40]

p2M=x_max[40:80]


p1m=p1m[::-1]
p1M=p1M[::-1]

#this is the lenghts Xn2

pm=np.abs(p1m)+np.abs(p2m)
pM=np.abs(p1M)+np.abs(p2M)

pm=pm*0.001

pM=pM*0.001

print(pm)
print(pM)
```



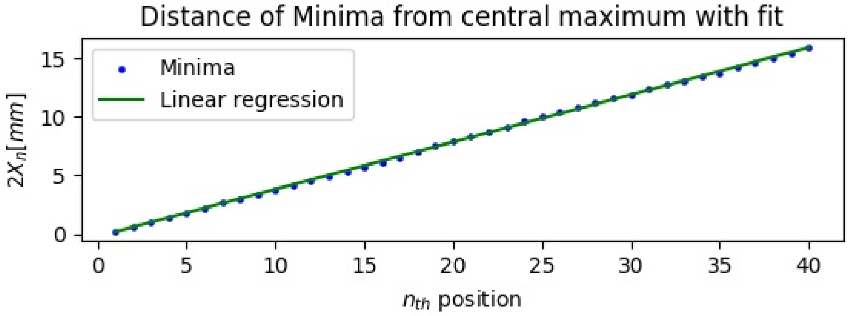

```
[ 0.203  0.595  1.022  1.421  1.827  2.212  2.646  3.017  3.318  3.752
  4.095  4.522  4.942  5.348  5.719  6.104  6.531  7.035  7.483  7.882
  8.302  8.722  9.163  9.59  10.08  10.437 10.801 11.158 11.592 11.914
 12.425 12.754 13.048 13.447 13.762 14.175 14.602 14.994 15.407 15.876]
[ 0.196  0.581  0.973  1.365  1.778  2.142  2.597  3.01   3.339  3.654
  4.123  4.557  4.963  5.327  5.677  6.041  6.573  6.923  7.322  7.798
  8.232  8.708  9.1    9.527  9.954 10.346 10.717 11.046 11.389 11.76
 12.215 12.684 13.034 13.419 13.748 14.266 14.693 15.113 15.4   15.792]
```

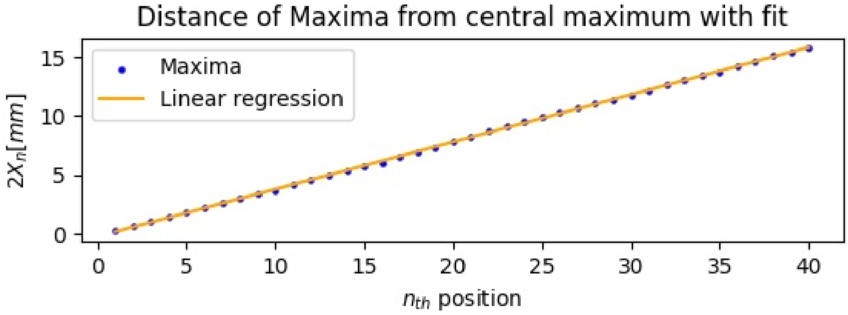
In [175…

```
l= 0.000636 #[mm]

#set focal distance:

f=300.8 #[mm]

n= np.array(list(range(1, 41)))


res_m = stats.linregress(n, pm)
res_M = stats.linregress(n, pM)



fig, (ax1, ax2) = plt.subplots(2)
fig.subplots_adjust(hspace=0.8)

ax1.scatter(n, pm, color='blue', label= "Minima", s= 5)
ax1.plot(n, res_m.intercept + res_m.slope*n, label='Linear regression', color="green")
ax1.set(xlabel=r"$n_{th}$ position", ylabel=r"$2X_n [mm]$")
ax1.set_title('Distance of Minima from central maximum with fit')
ax1.legend()

ax2.scatter(n, pM, color='blue', label= "Maxima", s= 5)
ax2.plot(n, res_M.intercept + res_M.slope*n, label='Linear regression', color="orange")
ax2.set(xlabel=r"$n_{th}$ position", ylabel=r"$2X_n [mm]$")
ax2.set_title('Distance of Maxima from central maximum with fit')
ax2.legend()


slope=res_M[0]

#from our formula, slope= 2*f*l/b -> b=2*f*l/slope
```

```
g=2*f*l/slope

print(g)
```

```
0.949026570021332
```





Loading [MathJax]/jax/output/CommonHTML/fonts/TeX/fontdata.js