Code used for the experiment

Code to read data from the csv file and plot the Gaussian beam profile.

```
import pandas as pd
import math
import numpy as np
import pylab as plb
import matplotlib.pyplot as plt
from scipy.optimize import curve_fit
from scipy import asarray as ar,exp

read=pd.read_csv("./Downloads/Gaussian Beam - Rajnil_Analysis_Copy_D=20.csv")

reading=[]
distance=[]

for i in read["TR"]:
    distance.append(i)
print(distance)

for i in read["Current "]:
    reading.append(i)
print(reading)

x_1 = np.array(distance)
y_1 =np.array(reading)

lower_bounds = [0, -np.inf, 0]  # Lower bounds: [a, x0, sigma]
upper_bounds = [np.inf, np.inf, np.inf]  # Upper bounds

n = len(x_1)                            #the number of data
mean = sum(x_1 * y_1) / sum(y_1)                  #note this correction
sigma=np.sqrt(sum(y_1 * (x_1 - mean)**2) / sum(y_1))  #note this correction
print(sigma,mean)


def gaus(x_1,a,x0,sigma):
    return a*exp(-(x_1-x0)**2/(2*sigma**2))

popt,pcov = curve_fit(gaus,x_1,y_1,p0=[25,mean,sigma],bounds=(lower_bounds, upper_bounds))


plt.plot(x_1,y_1,'o-',label='data')
plt.axhline(y=np.max(y_1)*0.135, color = 'r', linestyle = '-')
plt.plot(x_1,gaus(x_1,*popt),label='fit')
plt.title("D=20cm, Reading 1, X-axis")
plt.legend()
plt.xlabel('Distance')
plt.ylabel('Current')
plt.show()

print(np.max(y_1)*0.135)
print(*popt)
print(pcov)
```

Code for the bootstrap re-sampling method used to calculate the error in the mean and standard deviation of the Gaussian beam radius.

```
# Number of bootstrap iterations
n_bootstrap = 1000
```

```
# Initialize arrays to store bootstrapped parameter values
bootstrapped_means = []
bootstrapped_sigmas = []

# Perform bootstrap resampling
for _ in range(n_bootstrap):
    # Generate random indices for resampling with replacement
    indices = np.random.randint(0, len(x_1), len(x_1))
    x_bootstrap = x_1[indices]
    y_bootstrap = y_1[indices]

    # Fit the Gaussian model to the bootstrapped data
    popt_bootstrap, _ = curve_fit(gaus, x_bootstrap, y_bootstrap, p0=[25, np.mean(x_bootstrap),
np.std(x_bootstrap)])

    # Store the bootstrapped parameter values
    bootstrapped_means.append(popt_bootstrap[1])
    bootstrapped_sigmas.append(popt_bootstrap[2])

# Calculate the standard deviation of the bootstrapped parameter distributions
mean_error = np.std(bootstrapped_means)
sigma_error = np.std(bootstrapped_sigmas)

print("Error in mean:", mean_error)
print("Error in standard deviation:", sigma_error)
```

Code to obtain the intersection points of the Gaussian beam profile with the $1/e^2$ intensity level.

```
import numpy as np
from scipy.optimize import fsolve

# Define the Gaussian function
def gaus(x, a, x0, sigma):
    return a * np.exp(-(x - x0)**2 / (2 * sigma**2))

# Parameters from the Gaussian fit
a, x0, sigma = popt  # Replace popt with your actual parameter values

# Define the straight line equation
def straight_line(x):
    return np.max(y_1) * 0.135

# Define the equation to find the intersection points
def equation_to_solve(x):
    return gaus(x, a, x0, sigma) - straight_line(x)

# Initial guesses for the intersection points (where the Gaussian is likely to intersect the
line)
x_guesses = [x0 - 2 * sigma, x0 + 2 * sigma]

# Find the roots (intersection points) using fsolve
intersection_points = fsolve(equation_to_solve, x_guesses)

print("Intersection points:", intersection_points)
```

Disclaimer: The code is not optimised for general use and some individual values need to be enetered by hand. It is preferable to break this down into cells in a Jupyter notebook and run it cell by cell.