This is not a straightforward problem and you will need to think about what you are doing and understand what you are doing to solve it. It'll teach you, in addition, how to read excel files in python.

Often times, curves and surfaces are not defined using equations but simply as sampled points (like in CAD). For example:

Equation Form:

The quadratic function is given by:

$$y = ax^2 + bx + c$$

For our example, let's use the specific equation:

$$y = x^2$$

This represents a simple parabola that opens upward.

Sampled Points:

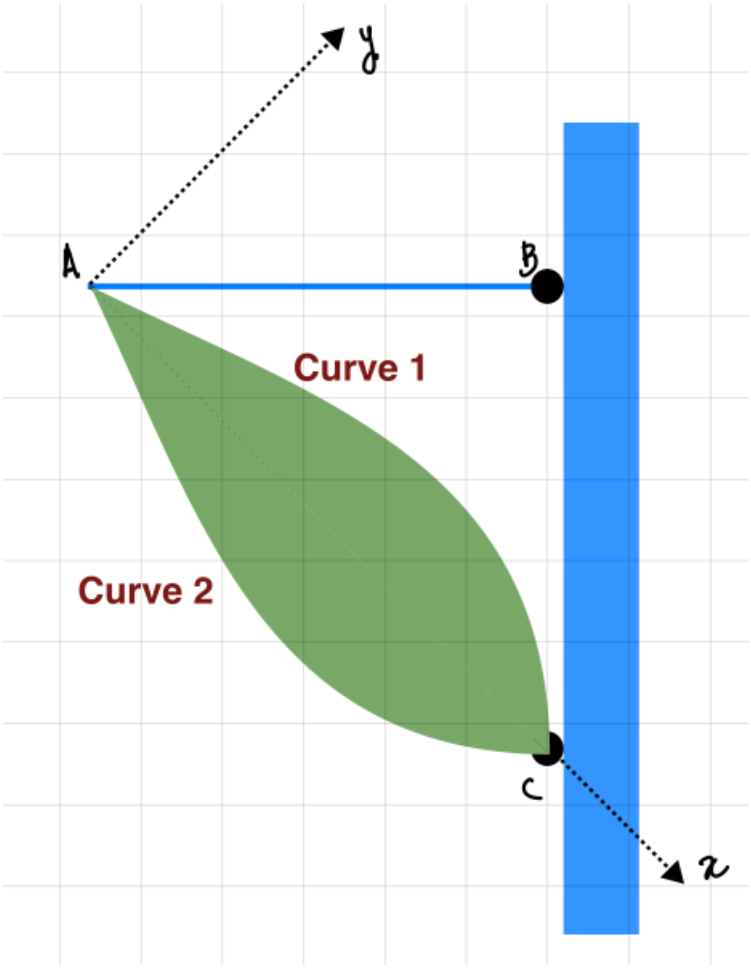
To represent this curve as sampled points, we'll compute the y-values for a range of x-values.

Let's take x-values from -5 to 5 in increments of 1.

x	y
-5	25
-4	16
-3	9
-2	4
-1	1
0	0
1	1
2	4
3	9
4	16
5	25

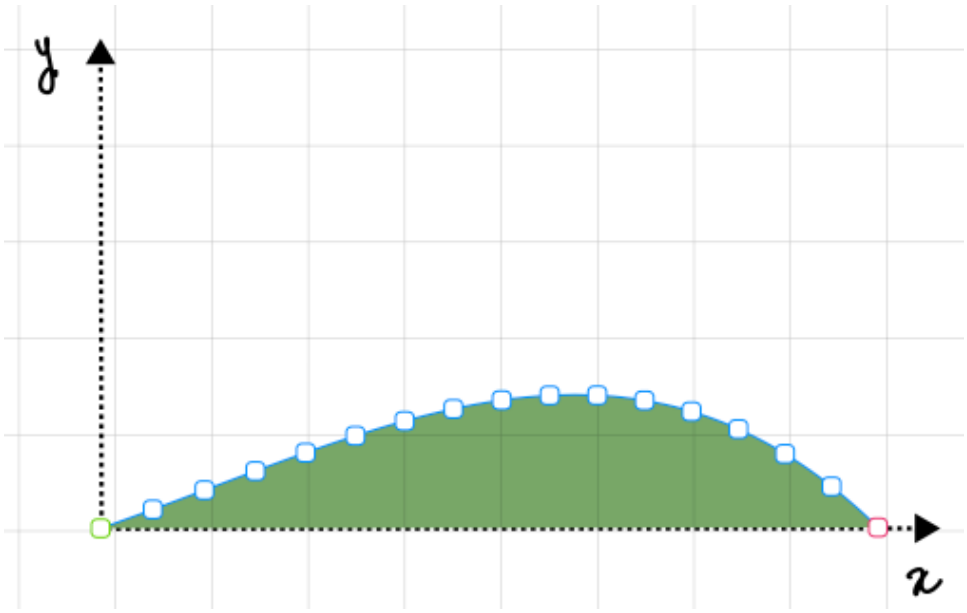
The above represents a sampled form of the original curve y=x^2.

Let’s now consider the following equilibrium problem:



You have to find the tension in the cable AB. The green region is a single rigid body which is defined by the two curves which are mirror-symmetric about the x axis. Its mass can be found by calculating its area A and multiplying it with the density d (assuming that the thickness inside the plane is also 1). Take d=10 and, therefore, mass=(10*A) Newtons.

AC is 1.618 meters and AB is 1 meters. Angle ABC is a right angle. In this problem you are not given the equations of curves 1 and 2 but you are given sampled points for curve 1 (curve 2 is simply a mirror image of curve 1 about the x axis). To be clear, if you rotate the x-y axis then the curve 1 (along with a diagrammatic representation of sampling) looks like this:

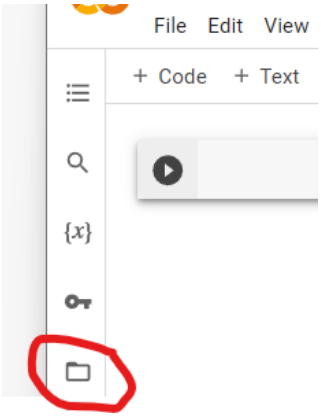


The sampled curve is provided in the excel file ‘sampled_points.xlsx’ where you are provided the (x,y) coordinates of the sampled points along the curve. You can open it and see what the data looks like.

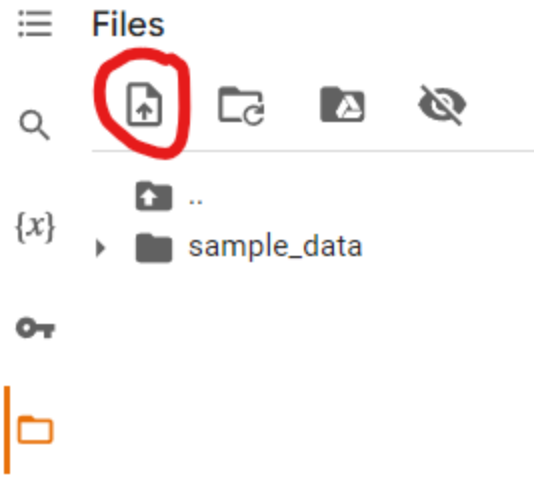
Here are the step by step instructions for solving this problem:

Upload the excel file to colab:

Click this in the colab file



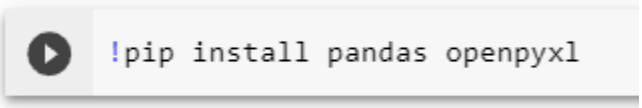
And then click this:



In order to upload the excel file to the colab session.

Install the libraries pandas and openpyxl in colab:

In the first cell of the colab file run this command to install pandas and openpyxl in the Colab environment:



*The **pandas** library is an open-source data manipulation and analysis tool for Python, renowned for its high-performance and ease of use. It introduces two main data structures: the **Series**, a one-dimensional labeled array, and the **DataFrame**, a two-dimensional table with heterogeneously-typed columns akin to a spreadsheet. Pandas excels at loading, manipulating, and exporting data from various formats such as CSV, Excel, and databases. The library offers robust functions for data cleaning, handling missing values, and transforming datasets. For data analysis, pandas provides functions for aggregation, grouping, and summarization of data. It features specialized support for time series data, enabling operations like date ranges and moving window statistics. Leveraging NumPy for fast computation, pandas ensures efficient performance even with large datasets. It seamlessly integrates with other Python libraries such as Matplotlib and Seaborn for visualization, and statsmodels and scikit-learn for statistical and machine learning tasks. Consequently, pandas has become a staple in the toolkit of data scientists, analysts, and developers working with data in Python. In essence, pandas simplifies and accelerates the end-to-end data analysis workflow.*

Read the contents of the excel file into numpy arrays using pandas and openpyxl:

Reading the Excel File:

Use the `read_excel` function from the `pandas` library to read the Excel file.

```
import pandas as pd

# Read the Excel file
df = pd.read_excel("sampled_points.xlsx", engine='openpyxl')
```

Here, `df` is a DataFrame, which is a 2-dimensional labeled data structure in pandas. Think of it as an in-memory Excel sheet with rows and columns.

Extracting x and y Variables:

Once you've read the Excel file into a DataFrame, you can extract the `x` and `y` columns to create the appropriate variables.

```
x = df['x'].values
y = df['y'].values
```

- `df['x']` extracts the column labeled 'x' from the DataFrame.
- `.values` converts the pandas Series (a single column of the DataFrame) into a numpy array.

Brief Explanations:

- The Excel file "sampled_points.xlsx" contains two columns: `x` and `y`, representing the sampled points of the function.
- By reading this file into a pandas DataFrame, you're loading these sampled points into a structured format in Python.
- Extracting the `x` and `y` columns from the DataFrame gives you the data in a format suitable for further processing or plotting.

Now, with the `x` and `y` variables created, you can use them for any further analysis, computations, or visualizations as needed.

Solve the problem:

- Find the area and then the mass of the green region
- Find the location of the centroid of the green region
- Solve the equilibrium problem

- Note that the only way to solve this problem is through numerical integration. Since you are not given a function f (as in previous integration problems), you will need to do the integration numerically. The process will be along the lines of the trapezoidal integration you have seen earlier but you need to think about what you are doing since **it won't be plug and play**. Think about calculating the integrals by breaking them down into thinner trapezoidal strips.

Numerical integration in python:

Numerically calculating the integral of a function over a given interval can be done using several methods, but one of the most common methods is the **trapezoidal rule**. Below is a step-by-step explanation and Python code using comments:

```python
import numpy as np

# Step 1: Define the function to be integrated
def f(x):
    return x**2  # Example: f(x) = x^2, you can replace this with any function

# Step 2: Define the interval of integration [x1, x2]
x1 = 0
x2 = 2

# Step 3: Choose the number of intervals/trapezoids.
# The more trapezoids you use, the more accurate your numerical integration will be.
n_intervals = 1000

# Step 4: Calculate the width of each trapezoid.
dx = (x2 - x1) / n_intervals

# Step 5: Use the trapezoidal rule to compute the integral.
# The idea behind the trapezoidal rule is to approximate the area under the curve
# using trapezoids and then sum up these areas.
integral = 0
for i in range(n_intervals):
    # left and right x-values of the current trapezoid
    x_left = x1 + i * dx
    x_right = x1 + (i + 1) * dx

    # average height of the function values at the left and right x-values
    avg_height = (f(x_left) + f(x_right)) / 2.0

    # area of the trapezoid is the average height times the width
    integral += avg_height * dx

# Step 6: Print the result
print(f"The numerical integral of f(x) from {x1} to {x2} is approximately {integral:.6f}")
```

```
The numerical integral of f(x) from 0 to 2 is approximately 2.666668
```

Submission requirements:

- A single pdf document should be submitted. It should include
  - your python code
  - The area and mass of the green region
  - The centroid of the green region in the x-y coordinate system
  - The tension in the cable