**Project Report**

**Numeric Computing – 102315**

**THE LENGTH OF A CURVE**

Syed Wajih Haider 9476

Fasih Amjad 9465

In this experiment, we will find the length of two curves generated from the same points – one curve is a polynomial interpolant and another one is a spline interpolant.

**Motivation behind the experiment**: In 1901, Runge conducted a numerical experiment to show that higher order interpolation is a bad idea. It was [shown](https://autarkaw.wordpress.com/2008/06/14/higher-order-interpolation-is-a-bad-idea/) that as you use higher order interpolants to approximate f(x)=1/(1+25x2) in [-1,1], the differences between the original function and the interpolants becomes worse. This concept also becomes the basis why we use splines rather than polynomial interpolation to find smooth paths to travel through several discrete points.

**What do we do:** A flexible curve (see Figure) made of lead-core construction with graduations in both millimeters and inches is provided? The student needs to draw a curve similar in shape to the Runge’s curve on the provided graphing paper as shown. It just needs to be similar in shape – the student can make the x-domain shorter and the maximum y-value larger or vice-versa. The student just needs to make sure that there is a one-to-one correspondence of values.

**Tools using: Python**

**Project Code:**

import numpy as np

import matplotlib.pyplot as plt

x\_pts = [-4.0,-2.5,-1.5,0,1.25,1.75,2.25,3.5]

#x\_pts =np.linspace(0, 2\*np.pi, 10) # 10 equidistant x coords from 0 to 10

#y\_pts =np.sin(x\_pts)

y\_pts = [0,0.75,2.75,4,3.5,2.25,1.00,0]

print("Please enter the original length of the wire")

lent = float(input())

#x\_pts = []

#print("Enter 9 points of x axis")

#for i in range (8):

# x\_pts.append(float(input()))

#

#y\_pts = []

#print("Enter 9 points of x axis")

#for i in range (8):

# y\_pts.append(float(input()))

x\_vals =np.linspace(-4, 2\*np.pi, 50) # 50 desired points

y\_vals =np.interp(x\_vals, x\_pts, y\_pts)

plt.plot(x\_pts, y\_pts, 'o') # plot known data points

plt.plot(x\_vals, y\_vals, '-x') # plot interpolated points

plt.show()

import numpy as np

from scipy import interpolate

import matplotlib.pyplot as plt

splines =interpolate.splrep(x\_pts, y\_pts)

x\_vals1 =np.linspace(-4, 2\*np.pi, 50) # 50 desired points

y\_vals1 =interpolate.splev(x\_vals, splines)

#plt.plot(x\_pts1, y\_pts1, 'o') # plot known data points

#plt.plot(x\_vals1, y\_vals1, '-x') # plot interpolated points

#plt.show()

def length(x,y,x1,l):

a = []

for i in range (49):

if(x[i]>=x1[7]):

break

else:

t = (x[i+1] - x[i])\*(x[i+1] - x[i])

g = (y[i+1] - y[i])\*(y[i+1] - y[i])

a.append(np.sqrt(t+g))

# print(a)

result = 0

for i in range (len(a)):

result = result + a[i]

print(result)

print("Difference")

print(l-result)

plt.plot(x\_pts, y\_pts, 'o', x\_vals, y\_vals, '-x', x\_vals1, y\_vals1, '-x')

plt.show()

print("Length of Polynomial Interpolation")

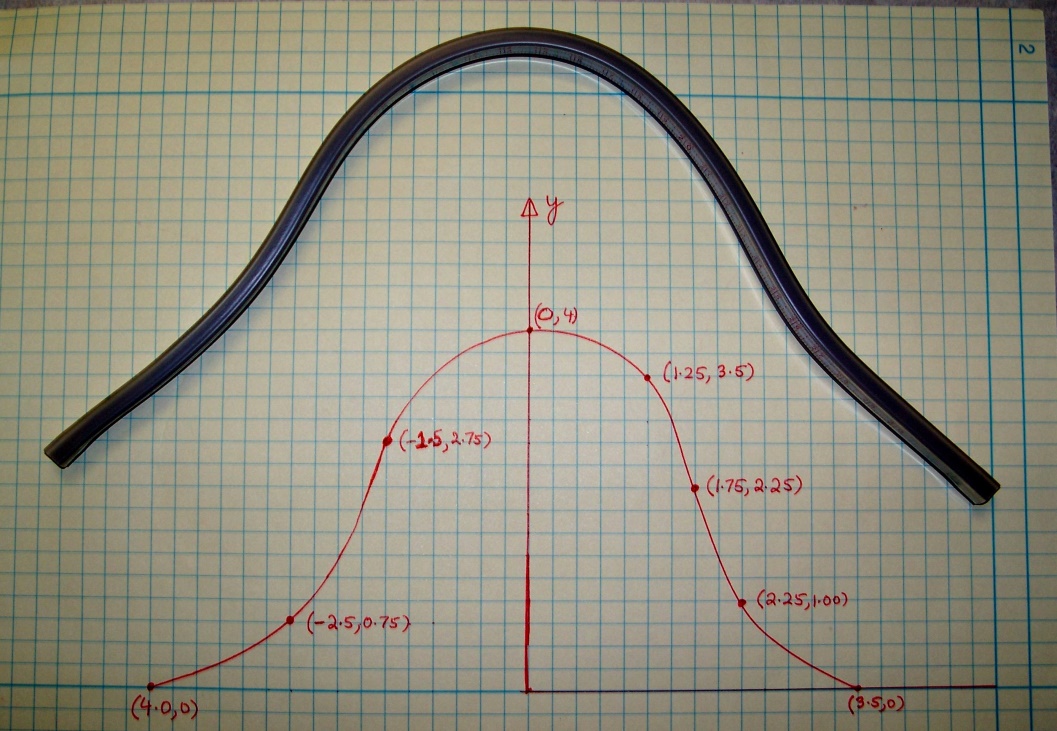
length(x\_vals,y\_vals,x\_pts,lent)

print("Length of Spline Interpolation")

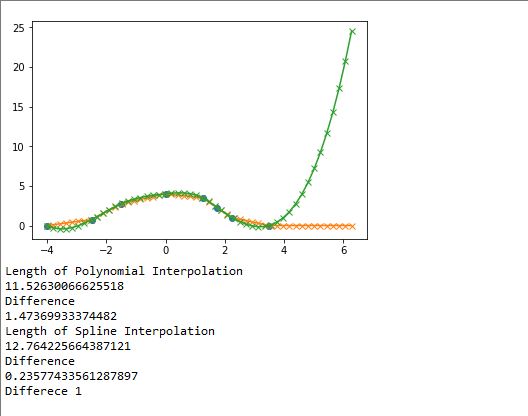
length(x\_vals1,y\_vals1,x\_pts,lent)

print()

**Actual Graph:**



**Screen Shots:**

****