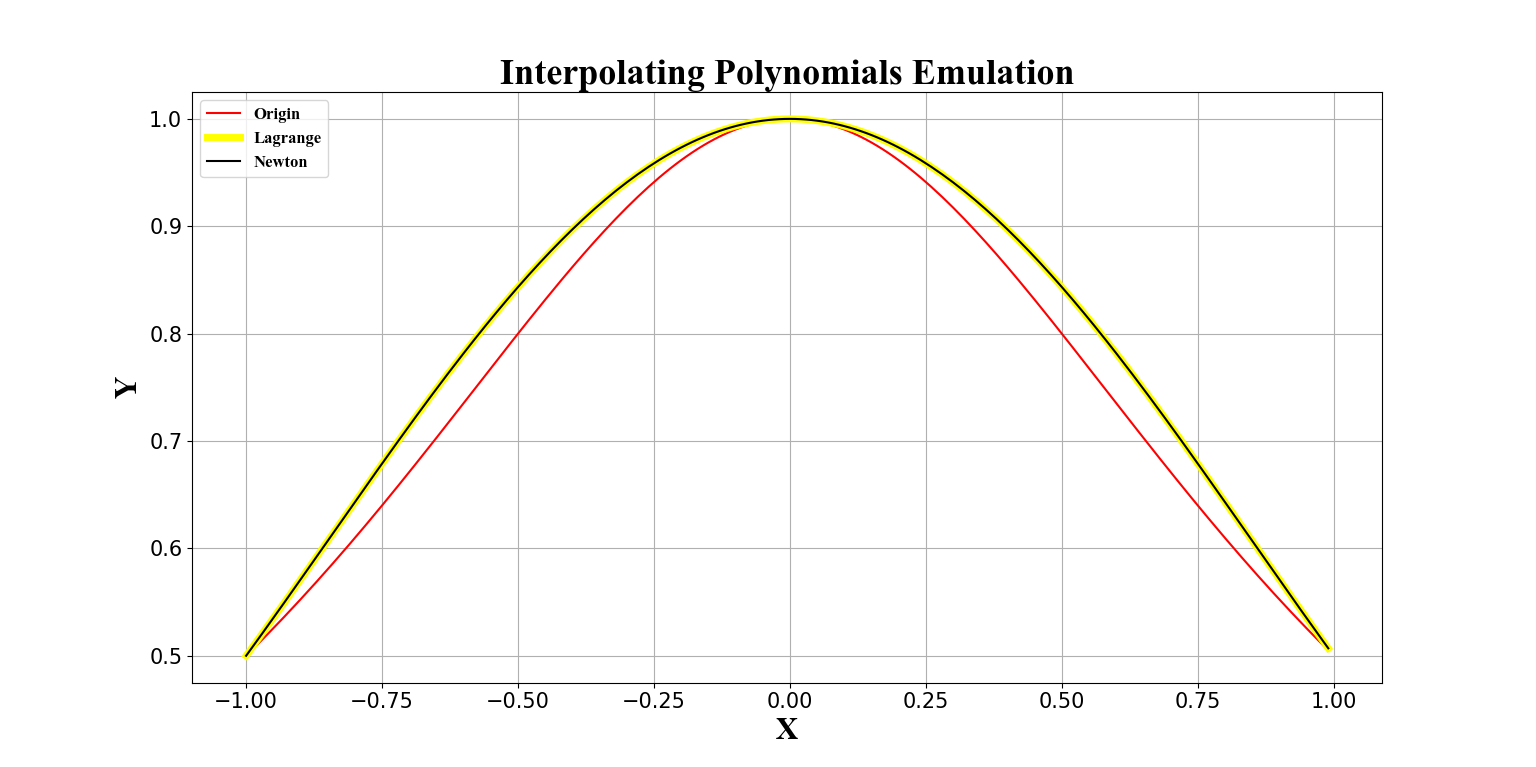
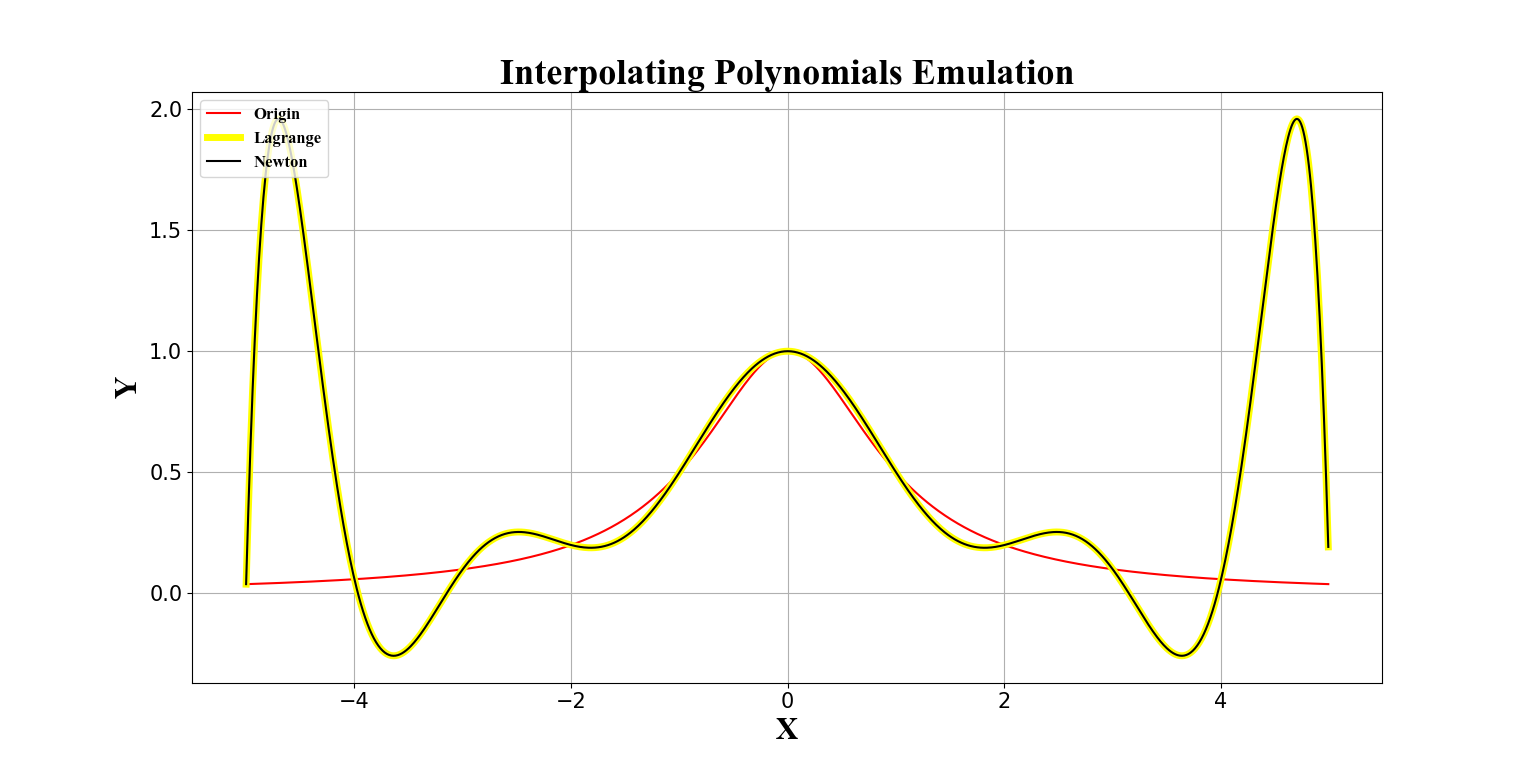
Given the Function f(x), Compute the interpolating polynomials based on the points

using both Lagrange and Newton interpolation algorithms. With the help of python for drawing the curves, we can surprisedly find that both of the Lagrange and Newton interpolation algorithms get the same polynomial, but it differs from the origin function in the different area.



Small range

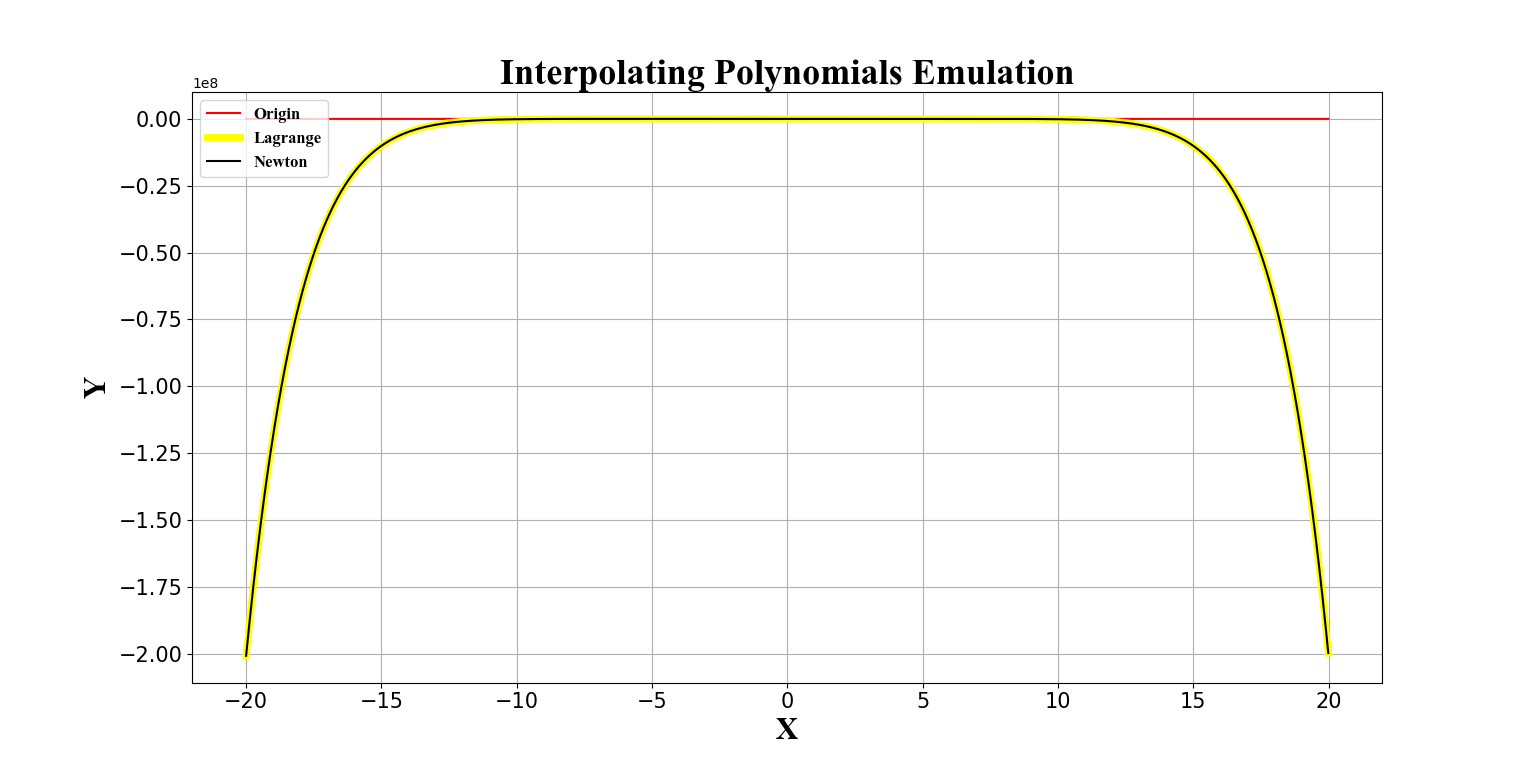
In the small range around the given points of x, especially the neighborhood of 0, the values from origin function is close to the polynomial function.



\

middle range

In the middle range, the values of polynomial function vibrate, but it vibrates around the origin function, especially when it is near the given values of x.



When the range enlarging, such as when it is smaller than the minimum or larger than the maximum value in the given x, the polynomial function goes farther away from the origin function.

Appendix: the code of python

import matplotlib.pyplot as plt

X=[-5.0,-4.0,-3.0,-2.0,-1.0,0.0,1.0,2.0,3.0,4.0,5.0]

Y=[0.0385,0.0588,0.1,0.2,0.5,1.0,0.5,0.2,0.1,0.0588,0.0385]

Z\_lagrange=[]

Z\_Newton=[]

size=len(X)

for i in range(size):

z=1.0

for j in range(size):

if j!=i:

z\*=X[i]-X[j]

Z\_lagrange.append(Y[i]/z)

Y\_tmp=Y

for i in range(size-1):

Z\_Newton.append(Y\_tmp[0])

for j in range(size-i-1):

Y\_tmp[j]=(Y\_tmp[j+1]-Y\_tmp[j])/(i+1)

Z\_Newton.append(Y\_tmp[0])

def Origin\_function(a):

return (1/(1+a\*a))

def Lagrange\_function(a):

all=1.0

ans=0.0

for i in range(size):

all=1.0

for j in range(size):

if i!=j:

all\*=(a-X[j])

ans+=all\*(Z\_lagrange[i])

return ans

def Newton\_function(a):

all=1.0

ans=0.0

for i in range(size):

ans+=all\*Z\_Newton[i]

all\*=(a-X[i])

return ans

x=[]

for i in range(0,200,1):

x.append(-1+0.01\*i)

#you can change the range here

#such as: for i in range(0,4000,1):

#x.append(-20+0.01\*i)

Origin=[]

Lagrange=[]

Newton=[]

for i in x:

Origin.append(Origin\_function(i))

Lagrange.append(Lagrange\_function(i))

Newton.append(Newton\_function(i))

l1, =plt.plot(x, Origin,linewidth = 1.5,c='red')

l2, =plt.plot(x, Lagrange,linewidth = 5,c='yellow')

l3, =plt.plot(x, Newton,linewidth = 1.5,c='black')

plt.title("Interpolating Polynomials Emulation", fontdict={'family' : 'Times New Roman', 'size' : 26})

plt.xlabel("X", fontproperties = 'Times New Roman',fontsize=22)

plt.ylabel("Y", fontproperties = 'Times New Roman',fontsize=22)

plt.tick\_params(axis='both', labelsize=15)

plt.legend(handles = [l1,l2,l3], labels = ['Origin','Lagrange','Newton'],loc = 'upper left', prop={'family' : 'Times New Roman', 'size' : 12})

plt.grid()

plt.show()