Задание:

1. Для функции заданной таблицей значений выполнить расчеты коэффициентов сглаживающих многочленов степеней 1, 2, 3 по методу наименьших квадратов и интерполяционного многочлена. Расчеты выполнить вручную либо с использованием математических программных пакетов типа MathCAD, microsoft mathematics и т.п.

2. Разработать программную реализацию построения интерполяционного многочлена с помощью формул Лагранжа и Ньютона; сглаживающих многочленов 1, 2, 3 степени по методу наименьших квадратов. Входной информацией для разработанной программы должна быть произвольная таблица значений функции. Выходная информация – графики функций. Кроме того, программа должна позволять отобразить график произвольного многочлена заданного набором коэффициентов. Стили отображения линий (штриховка, цвет, толщина и т.п.) должны позволить увидеть совпадающие графики.

3. Проверить совпадение графиков многочленов рассчитанных вручную с построенными программно.

Исходные данные:



Вычисление методом Ньютона:

f10=(y[1]-y[0])/(x[1]-x[0])= -3.5

f11=(y[2]-y[1])/(x[2]-x[1])= 1.5

f12=(y[3]-y[2])/(x[3]-x[2])= 4.5

f13=(y[4]-y[3])/(x[4]-x[3])= -7.0

f20=(f11-f10)/(x[2]-x[0])= 1.25

f21=(f12-f11)/(x[3]-x[1])= 0.75

f22=(f13-f12)/(x[4]-x[2])= -2.875

f30=(f21-f20)/(x[3]-x[0])= -0.083

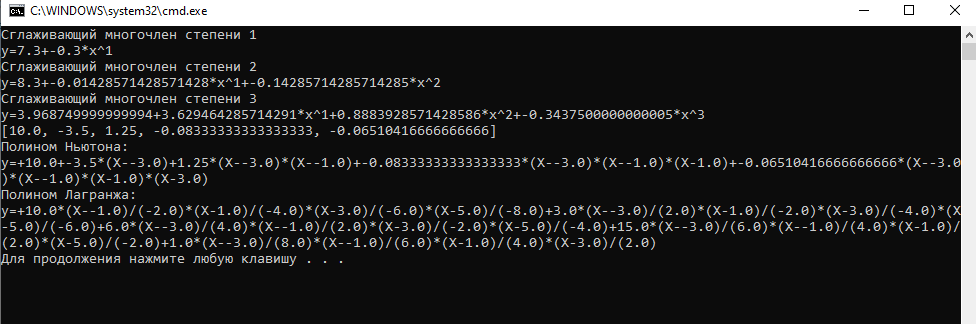
f31=(f22-f21)/(x[4]-x[1])= -0.604

f40=(f31-f30)/(x[4]-x[0])= -0.065

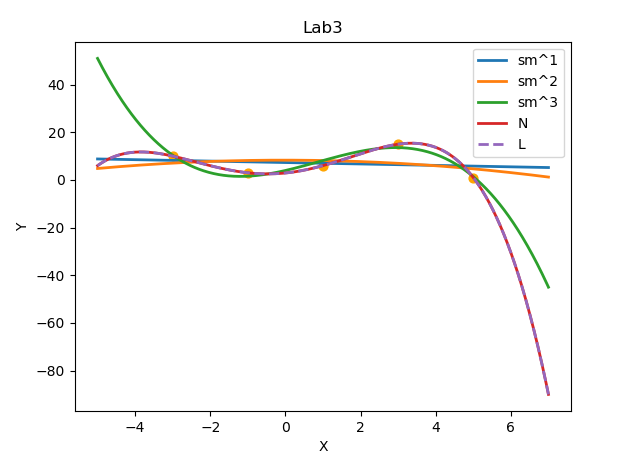
y(x)=10-3.5\*(x+3)+1.25\*(x+3)\*(x+1)- 0.083\*(x+3)\*(x+1)\*(x-1)-0.065\*(x+3)\*(x+1)\*(x-1)\*(x-3)

y(x)= -0.065 x^4 - 0.083 x^3 + 1.651 x^2 + 1.583 x + 2.914

Результаты программной реализации:



Графики:



Текст программы (Python):

(Для работы требуется наличие файла “tabl\_znach.txt” в каталоге программы, пакеты “matplotlib”, “nympy”, “sympy”)

import matplotlib.pyplot as plt

import numpy as np

from sympy import \*

#получение таблицы значений из файла

#в файле первая строка x, вторая y

tabl\_znach=[]

with open("tabl\_znach.txt") as file\_object:

for line in file\_object:

tabl\_znach.append(line.split())

x=[float(elem)for elem in tabl\_znach[0]]

y=[float(elem)for elem in tabl\_znach[1]]

#Параметры окна отображения графиков

plt.title("Lab3")

plt.xlabel("X")

plt.ylabel("Y")

legend = []

#Область отображения графиков

fx = np.linspace(x[0]-2,x[-1] + 2, 1000)

#метод наименьших квадратов

c=[]

for m in range(7):

temp=0

for i in range(len(x)):

temp+=x[i]\*\*m

c.append(temp)

d=[]

for j in range(4):

temp=0

for i in range(len(x)):

temp+=y[i]\*(x[i]\*\*j)

d.append(temp)

def slau(n):

m=np.array(n, dtype=np.float64)

for i in range(len(m)):

j=i+1

for j in range(len(m)):

if m[i][i]<m[j][i]:

m[i],m[j]=m.copy()[j],m.copy()[i]

for i in range(len(m)):

j=0

for j in range(len(m)):

if (i!=j):

k=m[j][i]/m[i][i]

m[j]=m[j]-k\*m[i]

a=[]

for i in range(len(m)):

a.append((m[i][len(m)]/m[i][i]))

return a

for s in range(3):

m=[]

for i in range(s+2):

m.append([])

for j in range(s+2):

m[i].append(c[j+i])

m[i].append(d[i])

a=slau(m)

sm=a[0].copy()

sm\_f=f'y={a[0]}'

for i in range(s+1):

sm+=a[i+1]\*(fx\*\*(i+1))

sm\_f+=(f"+{a[i+1]}\*x^{i+1}")

print(f"Сглаживающий многочлен степени {s+1}")

print(sm\_f)

plt.plot(fx, sm, linewidth=2)

legend.append(f"sm^{s+1}" % sm)

#отображение исходных точек

plt.scatter(x, y, color='orange', s=40)

#Полином Ньютона

n = len(x)

k = []

for i in range(n):

k.append(y[i])

for j in range(1, n):

for i in range(n-1, j-1, -1):

k[i] = float(k[i]-k[i-1])/float(x[i]-x[i-j])

print(k)

rez=0

form="y="

print("Полином Ньютона:")

for i in range(len(k)):

l=k[i]

form+=(f"+{k[i]}")

for j in range(i):

l\*=(fx-x[j])

form+=(f"\*(X-{x[j]})")

rez+=l

print(form)

plt.plot(fx, rez, linewidth=2)

legend.append(f"N" % rez)

#Полином Лагранжа

def lag(x,y,fx):

rez=0

form="y="

print("Полином Лагранжа:")

for i in range(len(x)):

l=y[i]

form+=(f"+{y[i]}")

for j in range(len(x)):

if (i!=j):

l\*=((fx-x[j])/(x[i]-x[j]))

form+=(f"\*(X-{x[j]})/({x[i]-x[j]})")

rez+=l

print(form)

return rez

L=lag(x,y,fx)

plt.plot(fx, L,"--", linewidth=2)

legend.append("L" % L)

#Отображение графиков

plt.legend(legend)#, loc="upper left")

plt.show()