# ЛАБОРАТОРНАЯ РАБОТА 4

**Приближение функций**

Вариант 20

Студент: Маркаров М.Г

Преподаватель: доц. Мамонтов

\

**Задача 4.1**. Функция *y=f(x)* задана таблицей значений  в точках . Используя метод наименьших квадратов (МНК), найти многочлены степеней m=1,2,…5 и определить многочлен с минимальным значением СКО: .

ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ:

1. Задать векторы *x* и *y* исходных данных.

2.Для нахождения коэффициентов многочлена составить нормальную систему метода наименьших квадратов и решить ее с помощью встроенной функции Python.

3. Для каждой степени *m*=0,1,2,...,5 найти многочлены *Pm* (коэффициенты многочленов) по методу наименьших квадратов. Вычислить соответствующие им значения . Выбрать наилучший многочлен.

4. На одном чертеже построить графики многочленов *Pm*, *m*=0,1,2,..., *m*\*, и точечный график исходной функции.

5.Результаты оформить в виде таблицы

Решение задачи 4.1:

1.

x=np.array([-0.7, -0.41, -0.2, 0.17, 0.46, 0.75, 1.04, 1.33, 1.62, 1.91, 2.2])

y=np.array([-12.917,3.619,9.586,7.949,1.543,-8.057,-16.150,-20.562,-17.720,-6.200,18.115])

2.

import numpy as np

def genumsys(m,x, y):

C=np.zeros((m+1, m+1))

b=np.zeros(m+1)

for k in range (m+1):

b[k]=np.sum(y\*x\*\*k)

for j in range (m +1):

C[k,j] = np.sum(x\*\*(k+j))

return C, b

for i in range (0,5):

C , b = genumsys(i,x,y)

a.append(np.linalg.solve(C,b))

print(a)

Результат работы программы:

[array([-3.70854545]), array([-2.69035863, -1.37087577]), array([-4.27961624, -9.19001741, 5.26149086]), array([ 9.99953011, -4.98864038, -40.00991299, 19.98835307]), array([ 10.04735338, -5.18620548, -40.18700388, 20.40108692, -0.13752325])]

Выводит массивом массивов.

3.Многочлены Pm и СКО

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

x=np.array([-0.7, -0.41, -0.2, 0.17, 0.46, 0.75, 1.04, 1.33, 1.62, 1.91, 2.2])

y=np.array([-12.917,3.619,9.586,7.949,1.543,-8.057,-16.150,-20.562,-17.720,-6.200,18.115])

def genumsys(m,x, y):

C=np.zeros((m+1, m+1))

b=np.zeros(m+1)

for k in range (m+1):

b[k]=np.sum(y\*x\*\*k)

for j in range (m +1):

C[k,j] = np.sum(x\*\*(k+j))

return C, b

def P(x , a):

r = 0

for i in range (len(a)):

r+=a[i]\*x\*\*i

return r

def sigma(x , y, a):

s = 0

for i in range(len(x)):

s+=(P(x[i], a) - y[i])\*\*2

s/=len(x)

s =np.sqrt(s)

return s

a = []

s = []

for i in range (0,5):

C , b = genumsys(i,x,y)

a.append(np.linalg.solve(C,b))

s.append(sigma(x,y,a[i]))

print(s)

Результат :

Массив СКО для m=0,1,...5

[12.117769486949443, 12.0513513775715, 11.400897885013219, 0.24386654414761294, 0.2362472188019741, 0.2133771813315586]

4. График всех Pm

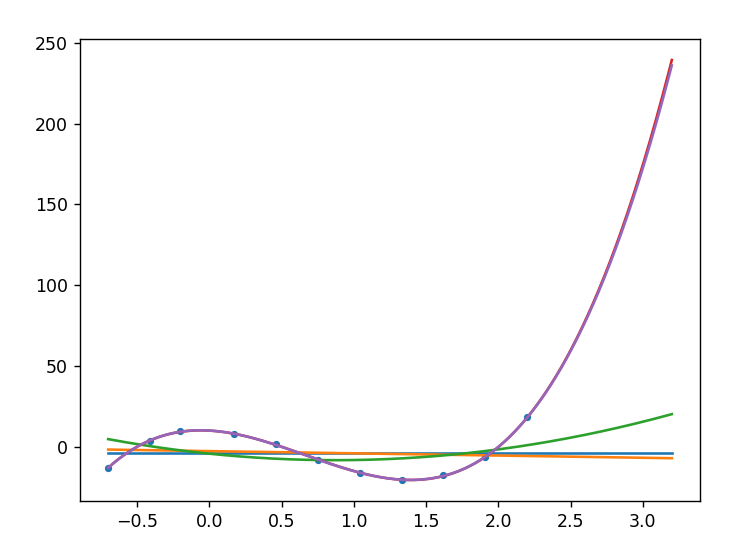
plt.scatter(x, y, 10)

x1 = np.linspace(-0.7,3.2,1000)

for i in range (0,5):

plt.plot(x1,P(x1,a[i]))

plt.show()

Имеем: 

5. Составим таблицу всех Pm и их СКО

|  |  |  |
| --- | --- | --- |
| Степень многочлена | Вид многочлена | СКО |
| m=0 | P0(x)= -3.7085x |  |
| m=1 | P1(x)= -2.6903 -1.3708x | 12.0513 |
| m=2 | P2(x)=-4.2796-9.1900x+5.2624x2 |  |
| m=3 | P3(x)=9.9995-4.9886-40.0099x2 +19.9883x3 |  |
| m=4 | P4(x)=10.0473-5.1862x-40.1870x2+20.4010x3-0.1375x4 |  |
| m=5 | P5(x)=10.1050-4.7053x-40.7911x2+19.4917x3+1.0595x4 -0.3227x5 | 0.2133 |

**Задача 4.2**. Функция *y=f(x)* задана на отрезке [a,b].Выполняется приближение функции интерполяционными многочленами с заданной точностью.

a) L– многочленом Лагранжа; b) S– кубическим сплайном (встроенная функция)

Требуется для каждого случая определить количество узлов, необходимых для достижения точности .

### ПОРЯДОК РЕШЕНИЯ ЗАДАЧИ

1. Составить подпрограмму вычисления массива значений функции  в n+1 точке отрезка , n- произвольный параметр, сетка равномерная c шагом таблицы h=(b-a)/n.

2. Составить подпрограммы, выполняющие интерполирование функции с помощью многочленов L(x), S(x) . 3.Составить подпрограмму вычисления погрешности приближения функции интерполяционными многочленами RL(x)= RS(x)=.

4.Задавая последовательно значения степени многочлена n=1,2,….построить графики погрешностей каждого многочлена. По графикам определить максимальную величину полученной погрешности приближения каждым способом. Если точность не достигнута, увеличить число узлов интерполяции.

5.Составить отчет по задаче.

Решение задачи 4.2

П.1+2.

Отрезок [0,5]

Программа дает на выход график f(x) по точкам и L(x):

import numpy as np

import matplotlib

import matplotlib.pyplot as plt

def f(x):

return x\*\*2 \* np.cos(4\*x)

def l(x,xft,i):

P = 1

for j in range (len(xft)):

if i != j :

P\*=(x-xft[j])/(xft[i]-xft[j])

return P

def L(x , xft, yft):

s = 0

for i in range (len(xft)):

s += yft[i]\*l(x,xft,i)

return s

n = 10

xft = np.linspace(0,5,n)

yft = f(xft)

print(L(3,xft,yft))

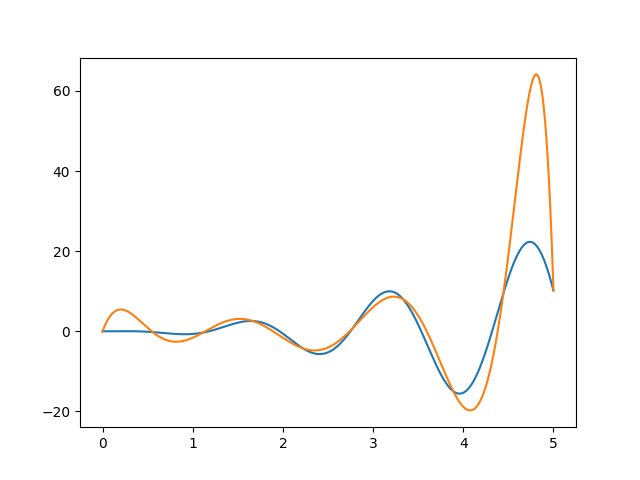
x1=np.linspace(0,5,1000)

plt.plot(x1,f(x1))

plt.plot(x1,L(x1,xft,yft))

plt.show()

Результат работы:



Подключим модуль scipy и воспользуемся функцией CubicSpline

from scipy.interpolate import CubicSpline

spline = CubicSpline(xft,yft)

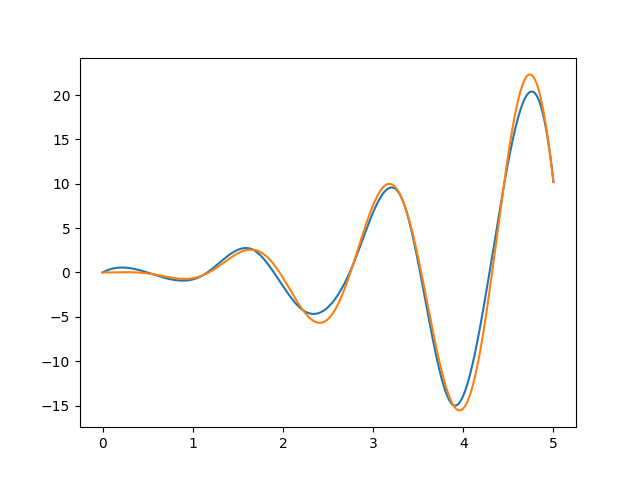
plt.plot(x1,spline(x1))

plt.plot(x1,f(x1))

plt.show()

Результат: график кубического сплайна S(x) и функции f(x) по точкам:

На данном этапе уже заметно различие между приближением многочленом Лагранжа и сплайном



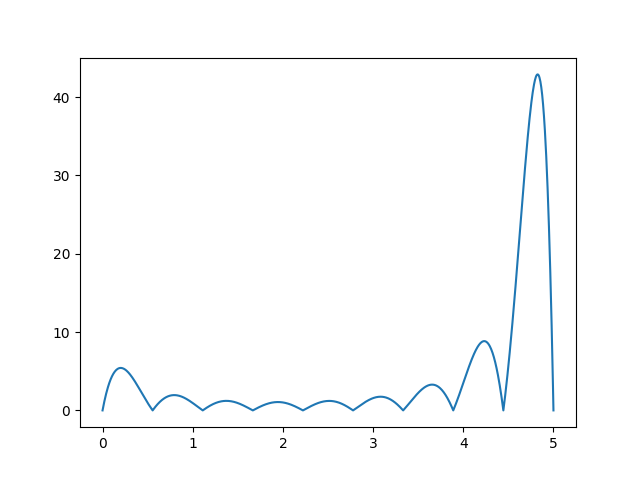
Пункт 3. Погрешности приближений:

1. Погрешность L(x):

plt.plot(x1,abs(f(x1)-L(x1,xft,yft)))

plt.show

Результат: Максимальная погрешность равна 44 из-за раскачки

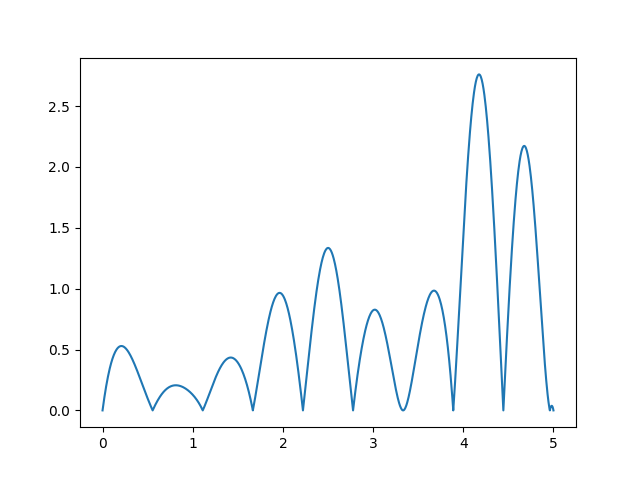


1. Погрешность S(x):

plt.plot(x1,abs(f(x1)-spline(x1)))

plt.show

Результат: Максимальная погрешность равна 2.9



**Задача 4.3.** Зависимость между величинами *x* и *y* описывается функцией *y=f(x, a, b),* где *a* и *b* – неизвестные параметры. Найти эти параметры, сведя исходную задачу к линейной задаче метода наименьших квадратов.

**УКАЗАНИЕ**. Свести исходную задачу к линейной задаче МНК можно, сделав подходящую замену переменных. Например, если исходная зависимость имеет вид , то прологарифмировав исходное равенство и введя новые переменные и , получаем задачу об определении коэффииентов линейной зависимости .

|  |  |
| --- | --- |
|  | |
| 1.5 | 0.99458 |
| 2.5 | 0.86240 |
| 3.5 | 0.75598 |
| 4.5 | 0.68569 |
| 5.5 | 0.63724 |
| 6.5 | 0.60212 |
| 7.5 | 0.57560 |
| 8.5 | 0.55489 |
| 9.5 | 0.53830 |
| 10.5 | 0.52472 |
| 11.5 | 0.51339 |

Замена t=arcsin(y) , s=1/x

Составим программу:

import numpy as np

x=[1.5,2.5,3.5,4.5,5.5,6.5,7.5,8.5,9.5,10.5,11.5]

y=[0.99458,0.86240,0.75598,0.68569,0.63724,0.60212,0.57560,0.55489,0.53830,0.52472,0.51339]

for i in range(len(y)):

y[i]=np.arcsin(y[i])

for i in range(len(x)):

x[i]=1/x[i]

invsum=0

for i in range(len(x)):

invsum+=pow(x[i],-1)

A=([len(x),sum(x)],[sum(x),invsum])

sum2=0

for i in range(len(y)):

sum2+=x[i]\*y[i]

B=([sum(y),sum2])

print(np.linalg.solve(A,B))

Результат: [0.75463429, 0.00690921] как массив [a,b]