# 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 массивом.

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

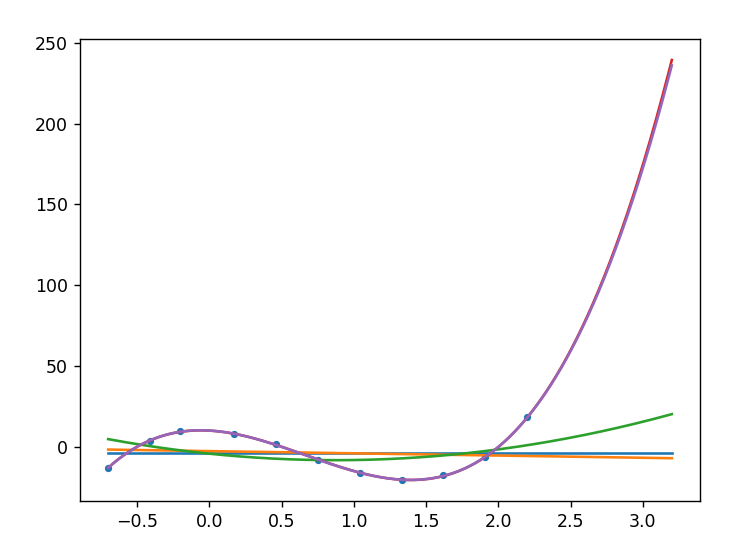
plt.scatter(x, y, 10)

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

1. Отрезок [0,5] , точность 0,00001.

**Задача 3.2.** Дана система уравнений Ax=b. Найти решение системы методом Зейделя с точностью .

1.Составить программу, реализующую метод Зейделя.

2. Найти решение системы из задачи 3.1 с заданной точностью . Использовать стандартный критерий окончания.

3. Методом Зейделя решить систему уравнений из таблицы 3.2 с точностью . Использовать стандартный критерий окончания.

4. Оформить отчет по задаче.

1. Метод Зейделя:

import numpy as np

def gauss\_seidel(A, b, x0, epsilon, max\_iterations):

n = len(A)

x = x0.copy()

for i in range(max\_iterations):

x\_new = np.zeros(n)

for j in range(n):

s1 = np.dot(A[j, :j], x\_new[:j])

s2 = np.dot(A[j, j + 1:], x[j + 1:])

x\_new[j] = (b[j] - s1 - s2) / A[j, j]

if np.allclose(x, x\_new, rtol=epsilon):

return x\_new

x = x\_new

return x

2. Произведем 5 итераций методом Зейделя:

seidelroot=gauss\_seidel(A, b, c, 10\*\*(-10), 5)

print(seidelroot)

Результат работы программы:

[ 2.00070264 4.81232225 -1.00951539 -6.99300058]

3. Решим другую систему уравнений( из таблицы 3.2)

N=20

m=20

R=0.1(N+20)=0.4

b[i]= 

A2=np.ndarray((20,20))

b2=np.ndarray(20)

c2=np.zeros(20)

for i in range(20):

for j in range(20):

A2[i,j]=0.4/np.log(i+j+2)

A2[i,i]=0.4/np.log(i+j+2) + 400/0.4

b2[i]= (-1)\*\*i \* np.sin(i) + 0.4

seidelroot2=gauss\_seidel(A2, b2, c2, 10\*\*(-10), 5)

print(seidelroot2)

Результат работы программы:

Вектор корней размерностью 20

[ 3.98731226e-04 -4.42845871e-04 1.30819516e-03 2.57724270e-04

-3.57950462e-04 1.35788085e-03 1.19522433e-04 -2.58032879e-04

1.38836801e-03 -1.31160227e-05 -1.45002195e-04 1.39904206e-03

-1.37521787e-04 -2.11010581e-05 1.38969335e-03 -2.51197999e-04

1.11198854e-04 1.36051200e-03 -3.51865536e-04 2.49253945e-04]

**Задача 3.3**. (не обязательно). Решить систему уравнений ** из задачи 3.2 методом релаксации с точностью **. Определить экспериментально параметр релаксации **, при котором точность ** достигается при наименьшем числе итераций. Построить графики зависимости числа итераций N от параметра релаксации**.

УКАЗАНИЕ. Параметр релаксации ** следует задавать из условия сходимости метода: **

Решение( при параметре релаксации 1.5 (произвольном) )

def sor\_solver(A, b, omega, initial\_guess, convergence\_criteria):

phi = initial\_guess[:]

residual = np.linalg.norm(np.matmul(A, phi) - b) #Initial residual

while residual > convergence\_criteria:

for i in range(A.shape[0]):

sigma = 0

for j in range(A.shape[1]):

if j != i:

sigma += A[i][j] \* phi[j]

phi[i] = (1 - omega) \* phi[i] + (omega / A[i][i]) \* (b[i] - sigma)

residual = np.linalg.norm(np.matmul(A, phi) - b)

return phi

SORroot=sor\_solver(A2,b2,1.5,c2,10\*\*(-10))

print(SORroot)

Результат:

Вектор корней размерностью 20

[ 3.98731224e-04 -4.42845872e-04 1.30819516e-03 2.57724269e-04

-3.57950463e-04 1.35788085e-03 1.19522432e-04 -2.58032879e-04

1.38836801e-03 -1.31160228e-05 -1.45002195e-04 1.39904206e-03

-1.37521787e-04 -2.11010581e-05 1.38969335e-03 -2.51197999e-04

1.11198854e-04 1.36051200e-03 -3.51865536e-04 2.49253945e-04]

Поиск оптимального параметра: