**МИНИСТЕРСТВО ОБРАЗОВАНИЯ РЕСПУБЛИКИ БЕЛАРУСЬ**

**УЧРЕЖДЕНИЕ ОБРАЗОВАНИЯ  
ГОМЕЛЬСКИЙ ГОСУДАРСТВЕННЫЙ ТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ИМЕНИ П. О. СУХОГО**

Факультет автоматизированных и информационных систем

Кафедра «Информационные технологии»

ОТЧЕТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №3

по дисциплине: «Численные методы математической физики**»**

на тему: «Разработка программ по методам аппроксимации функций»

Выполнил: студент гр. ИТП-22

Расшивалов Н.И.  
 Принял: доцент

Стародубцев Е.Г.

Гомель 2021

**Цель работы:** научиться разрабатывать алгоритмы численных методов и программное обеспечение для аппроксимации функций.

**ЗАДАНИЕ**

Разработать алгоритмы и написать программы, реализующие следующие методы интерполяции и аппроксимации функций:

1. с помощью канонического полинома;

2. с помощью полинома Лагранжа;

3. с помощью полинома Ньютона;

4. с помощью метода наименьших квадратов.

Данные опыта представлены таблицей значений Х и У. Подобрать к этим данным интерполяционные полиномы (канонический, Лагранжа, Ньютона) и выполнить аппроксимацию методом наименьших квадратов.

**Вариант 30.** Дана таблица удельного сопротивления Х (кН/м2) и производительности У (га/ч) посевного агрегата.

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| Х | 320 | 360 | 410 | 460 | 510 |
|  |  |  |  |  |  |
| У | 280 | 310 | 330 | 360 | 380 |
|  |  |  |  |  |  |
|  |  |  |  |  |  |

График исходной функции представлен на рисунке 1.

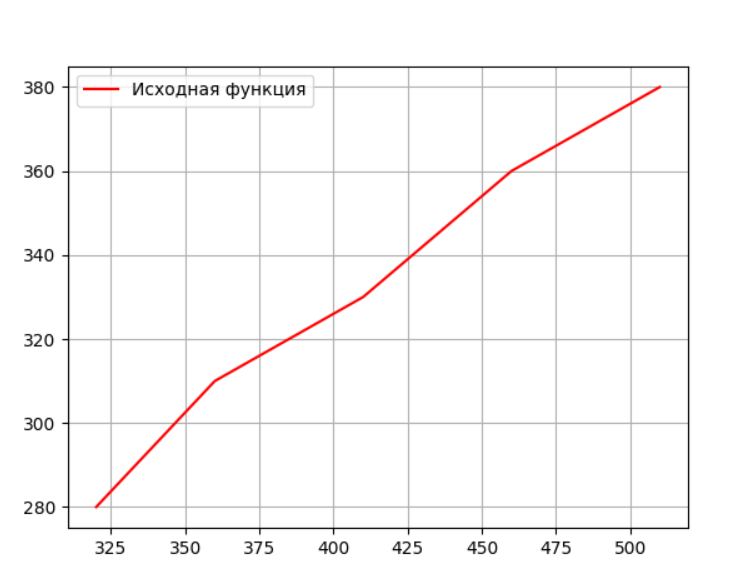


Рисунок 1 – График исходной функции

Интерполяция с помощью канонического полинома представлена на рисунке 2.

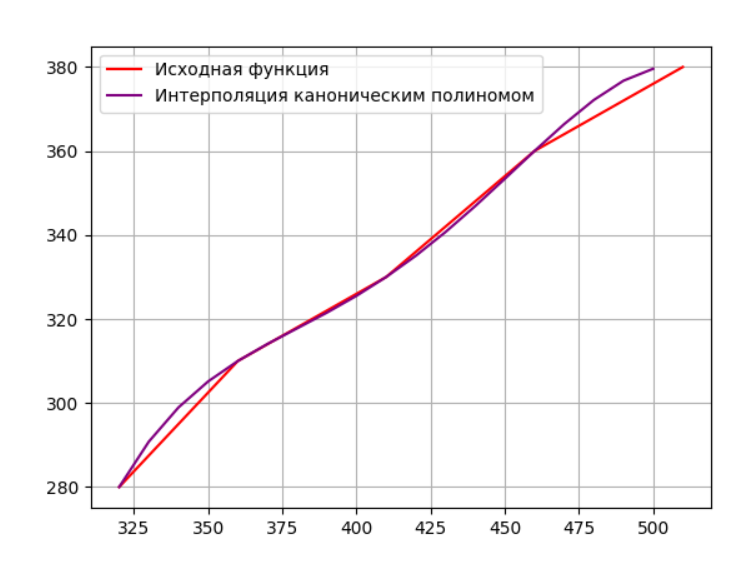


Рисунок 2 – Интерполяция с помощью канонического полинома

Интерполяция с помощью полинома Лагранжа представлена на рисунке 3.

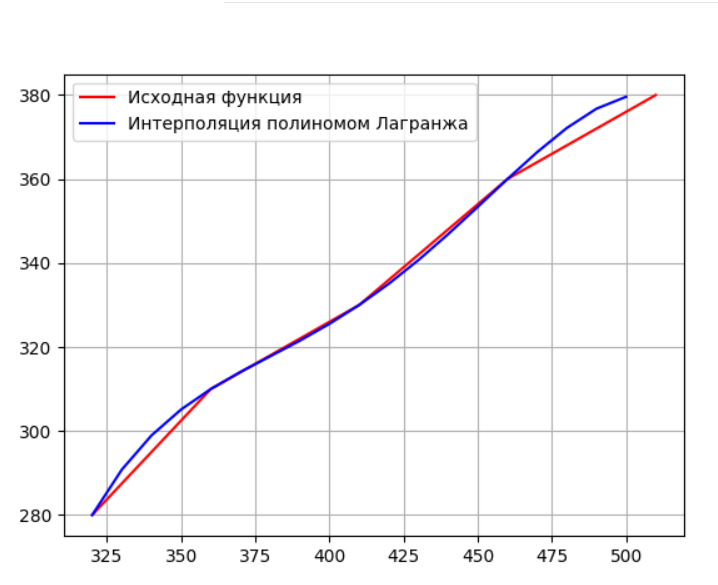


Рисунок 3 – Интерполяция с помощью полинома Лагранжа

Интерполяция с помощью полинома Ньютона представлена на рисунке 4.

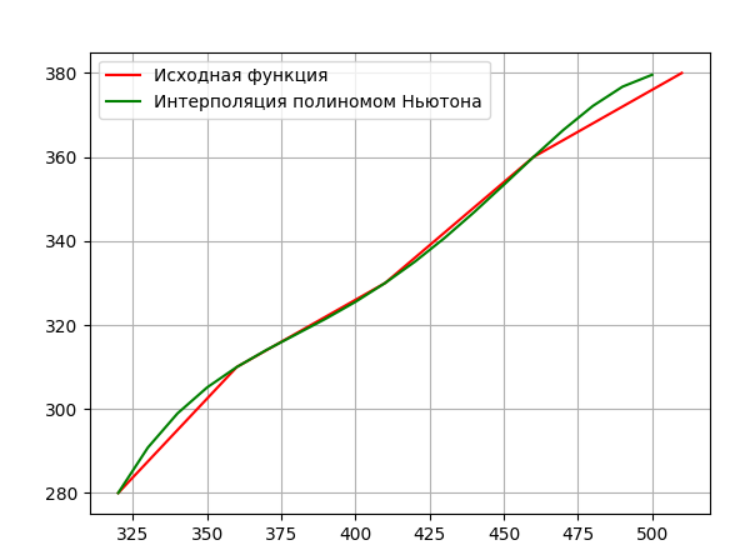


Рисунок 4 – Интерполяция с помощью полинома Ньютона

Аппроксимация функции методом наименьших квадратов представлена на рисунке 5.

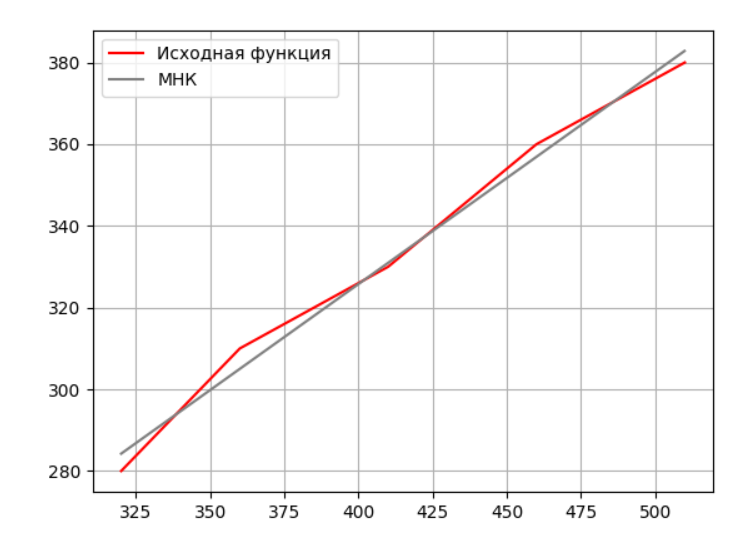


Рисунок 5 – Аппроксимация функции методом наименьших квадратов

Коэффициенты, полученные с помощью онлайн-калькулятора представлены на рисунке 6.

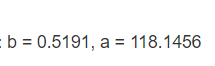


Рисунок 6 – Коэффициенты, полученные с помощью онлайн-калькулятора

Найденные коэффициенты линейной функции представлены на рисунке 7.



Рисунок 7 – Найденные коэффициенты линейной функции

**Вывод**: изучена разработка алгоритмов интерполяции и аппроксимации функций различными методами.

# **ПРИЛОЖЕНИЕ А**

**Графические схемы алгоритмов**



Рисунок A.1 – Графическая схема алгоритма интерполяции функции методом канонического полинома

matrix – расширенная матрица чисел

n– количество исходных значений

t – x в котором находится значение

i,j – счетчики циклов

A – коэффициенты СЛАУ

sum – значение функции в точке t



Рисунок A.2 – Графическая схема алгоритма интерполяции функции методом полинома Лагранжа

n– количество исходных значений

t – x в котором находится значение

l – функция от t удовлетворяющая определенным условиям

i,j – счетчики циклов

sum – значение функции в точке t



Рисунок A.3 – Графическая схема алгоритма интерполяции функции методом полинома Ньютона

A – диоганальные элементы C

C – матрица коэффициентов

sum – значение функции в точке t

n – количество значений



Рисунок A.4 – Графическая схема алгоритма аппроксимации функции методом наименьших квадратов

a0,a1 – коэффиценты функции f(x) = a0 + a1\*x

xsum ­– сумма значений x

ysum – сумма значений y

x2sum – сумма значений x в квадрате

xysum – сумма произведений x,y

# **ПРИЛОЖЕНИЕ Б**

**Листинг программы**

**#Метод канонического полинома**

#Находит массив значений в точке t от x0 до xn

def methodCanonic(x,y):

t = x[0]

xres = []

yres = []

while t < x[len(x)-1]:

yres.append(canonic(t,x,y))

xres.append(t)

t += 0.1

return xres,yres

#метод канонического полинома

def canonic(t,x,y):

n = len(x) #получение количества исходных значений

m = n+1 #размер расширенной матрицы

matrix = []

for i in range(0,n): #заполнение матрицы

matrix.append([])

for j in range(0,n):

matrix[i].append(pow(x[i],j))

for i in range(0,n): #заполнение свободных членов матрицы

matrix[i].append(y[i])

A = gausseSLAU(matrix) #решение СЛАУ

sum = 0 #значение функции в точке t

for i in range(0,n): #получение значения sum

sum += A[i] \* pow(t,i)

return sum

#решает слау методом Гаусса

def gausseSLAU(matrix):

n = len(matrix)

x = list(range(n))

i = 0

while i<n: #прямой ход (приведение матрицы к треугольному виду)

tmp = matrix[i][i]

j = n

while j >= i:

matrix[i][j] /= tmp #расчет коэффицента для k-ой строки

j -= 1

j = i+1

while j<n:

tmp = matrix[j][i]

k = n

while k >= i:

matrix[j][k] -= tmp\*matrix[i][k]

k -= 1

j += 1

i += 1

x[n - 1] = matrix[n - 1][n] #последний корень уже есть,с помощью его по очереди находятся остальные

i = n-2 #начало с предпоследней строки

while i >= 0: #обратный ход

x[i] = matrix[i][n]

j = i+1

while j<n:

x[i] -= matrix[i][j] \* x[j]

j += 1

i -= 1

return x

**#Метод полинома Лагранжа**

#Находит массив значений в точке t от x0 до xn

def methodLagrange(x,y):

t = x[0]

xres = [] #все значения t

yres = [] #все найденные f(t)

while t < x[len(x)-1]:

yres.append(lagrange(t,x,y))

xres.append(t)

t += 0.1

return xres,yres

#метод полинома Лагранжа

def lagrange(t,x,y):

n = len(x) #получение количества исходных значений

sum = 0 #значение функции в точке t

for i in range(0,n):

l = 1 # функция L(t)

for j in range(0,n):

if j != i:

l = l \* (t - x[j])/(x[i] - x[j]) #нахождение значения L(t)

sum += y[i]\*l

return sum

**#Апроксимация методом МНК**

def methodSmallestQuadrates(x,y):

n = len(x)

xsum = 0 #сумма значений x

ysum = 0 #сумма значений y

for i in range(0,len(x)):

xsum += x[i]

ysum += y[i]

x2sum = 0 #сумма квадратов x

xysum = 0 #сумма произведений x,y

for i in range(0,len(x)):

x2sum += x[i]\*x[i]

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

a1 = (n\*xysum - xsum\*ysum)/(n\*x2sum-xsum\*xsum) #коэффициент a1 функции y = a0 +a1\*x

a0 = ysum/n - a1\*xsum/n #коэффициент a0 функции y = a0 +a1\*x

print("a0:"+str(a0))

print("a1:"+str(a1))

xres = []

yres = []

t = x[0]

while t < x[len(x)-1]: #получение значений с помощью полученных коэффициентов

yres.append(a1\*t + a0)

xres.append(t)

t += 0.1

return xres,yres