**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 №4

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

на тему: «Разработка программ по методам численного интегрирования и дифференцирования»

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 численного интегрирования и дифференцирования.

**ЗАДАНИЕ**

1) Разработать алгоритмы и написать программы, реализующие следующие методы численного интегрирования:

а) прямоугольников (левых, правых, средних);

б) трапеций;

в) парабол (Симпсона).

Сравнить точность использованных методов.

2) Разработать алгоритмы и написать программы, реализующие следующие методы численного дифференцирования:

а) формулы на основе полинома Ньютона;

б) формулы на основе полинома Лагранжа.

Данные опыта представлены таблицей значений Х и У (дана функция f(x), заданная таблично). Вычислить по этим данным интеграл и производную функции f(x) (за пределы интегрирования взять наименьшее и наибольшее значение Х).

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

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

Решение определённого интеграла с помощью калькулятора представлено на рисунке 1.

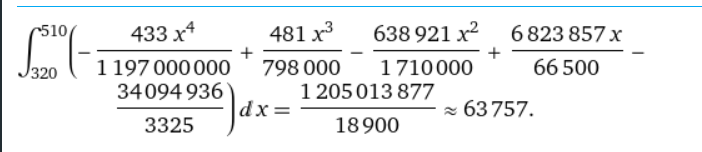


Рисунок 1 – Решение определённого интеграла с помощью калькулятора

Численное интегрирование с помощью метода левых прямоугольников представлено на рисунке 2.



Рисунок 2 – Численное интегрирование с помощью метода левых прямоугольников

Численное интегрирование с помощью метода правых прямоугольников представлено на рисунке 3.



Рисунок 3 – Численное интегрирование с помощью метода правых прямоугольников

Численное интегрирование с помощью метода средних прямоугольников представлено на рисунке 4.



Рисунок 4 – Численное интегрирование с помощью метода средних прямоугольников

Численное интегрирование с помощью метода трапеций представлено на рисунке 5.



Рисунок 5 – Численное интегрирование с помощью метода трапеций

Численное интегрирование с помощью метода Симпсона представлено на рисунке 6.



Рисунок 6 – Численное интегрирование с помощью метода Симпсона

Производная функции с помощью онлайн-калькулятора представлена на рисунке 7.

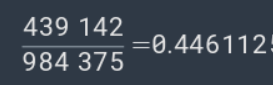


Рисунок 7 – Производная функции с помощью онлайн-калькулятора

Численное дифференцирование на основе полинома Ньютона представлено на рисунке 8.



Рисунок 8 – Численное дифференцирование на основе полинома Ньютона

Численное дифференцирование на основе полинома Лагранжа представлено на рисунке 9.



Рисунок 9 – Численное дифференцирование на основе полинома Лагранжа

**Вывод**: изучена разработка алгоритмов численного интегрирования и дифференцирования.

# **ПРИЛОЖЕНИЕ А**

**Графические схемы алгоритмов**

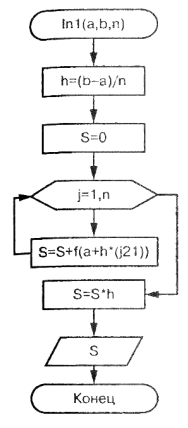


Рисунок A.1 – Графическая схема алгоритма левых прямоугольников

a – начало промежутка

b – конец промежутка

h – шаг

S – площадь

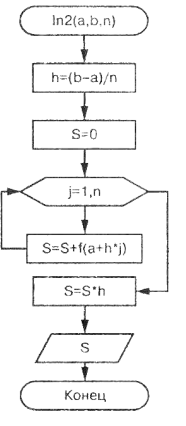


Рисунок A.2 – Графическая схема алгоритма правых прямоугольников

a – начало промежутка

b – конец промежутка

h – шаг

S – площадь

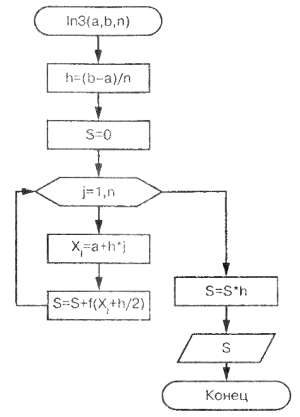


Рисунок A.3 – Графическая схема алгоритма правых прямоугольников

a – начало промежутка

b – конец промежутка

h – шаг

X – текущее значение

S – площадь

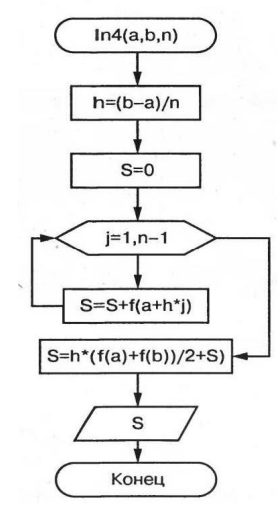


Рисунок A.4 – Графическая схема алгоритма трапеций

a – начало промежутка

b – конец промежутка

h – шаг

S – площадь

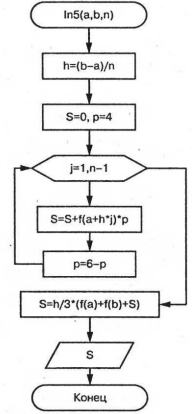


Рисунок A.4 – Графическая схема алгоритма Симпсона

a – начало промежутка

b – конец промежутка

h – шаг

S – площадь



Рисунок A.5 – Графическая схема алгоритма дифференцирования на основе полинома Ньютона

x,y – значения табличной функции

y1 – значение производной

t – точка в которой ищется производная

h – шаг

i – индекс точки близкой к t



Рисунок A.6 – Графическая схема алгоритма дифференцирования на основе полинома Лагранжа

x,y – значения табличной функции

y1 – значение производной

t – точка в которой ищется производная

h – шаг

i – индекс точки близкой к t

# **ПРИЛОЖЕНИЕ Б**

Листинг программы

import math

def main():

    x = [320,360,410,460,510] #значения x табличной функции

    y = [280,310,330,360,380] #значения y табличной функции

    rectangleLeft(x[0], x[len(x)-1], 1500,x,y)

    rectangleRight(x[0], x[len(x)-1], 1500,x,y)

    rectangleAverage(x[0], x[len(x)-1], 1500,x,y)

    trapeze(x[0], x[len(x)-1], 1500,x,y)

    Simpson(x[0], x[len(x)-1], 1500,x,y)

    difLagrange(x,y,359)

    difNewton(x,y,359)

def rectangleLeft(a,b,n,x,y): #метод левых прямоугольников

    h = (b-a) / n #шаг

    S = 0 #значение интеграла

    for j in range(0,n):

        S += lagrange(a + h\*j, x, y) #добавление значения функции в точке a + h\*j

    S = S \* h

    print("Метод левых прямоугольников: "+str(S))

def rectangleRight(a,b,n,x,y): #метод правых прямоугольников

    h = (b-a) / n #шаг

    S = 0 #значение интеграла

    for j in range(1,n+1):

        S += lagrange(a + h\*j, x, y) #добавление значения функции в точке a + h\*j

    S = S \* h

    print("Метод правых прямоугольников: "+str(S))

def rectangleAverage(a,b,n,x,y): #метод средних прямоугольников

    h = (b-a) / n #шаг

    S = 0 #значение интеграла

    for j in range(1,n+1):

        xi = a + h\*j

        S += lagrange(xi + h/2, x, y)

    S = S \* h

    print("Метод средних прямоугольников: "+str(S))

def trapeze(a,b,n,x,y): #метод трапеций

    h = (b-a) / n #шаг

    S = 0 #значение интеграла

    for j in range(1,n):

        xi = a + h\*j

        S += lagrange(xi + h/2, x, y) #добавление значения функции в точке xi + h/2

    S = h \* ((lagrange(a, x, y) + lagrange(b, x, y))/2 + S)

    print("Метод трапеций: "+str(S))

def Simpson(a,b,n,x,y): #метод Симпсона

    h = (b-a) / n #шаг

    S = 0 #значение интеграла

    p = 4

    for j in range(1,n):

        S += lagrange(a + h\*j, x, y) \* p

        p = 6 - p

    S = h/3 \* (lagrange(a, x, y) + lagrange(b, x, y) + S)

    print("Метод Симпсона: "+str(S))

def difLagrange(x,y,t): #производная на основе полинома Лагранжа

    x,y = methodLagrange(x,y) #интерполированные значения функции

    n = len(x) #количество значений

    h = (x[len(x)-1]-x[0])/n #шаг

    i = int((t-x[0])/h+h/2) #индекс точки близкой к t

    y1 = 0.0 #значение производной

    if i == 0:

        y1 = (-3\*y[0]+4\*y[1]-y[2])/(2\*h)

    elif i >0 and i < n:

        y1 = (-y[i-1]+y[i+1])/(2\*h)

    elif i == n:

        y1 = (y[n-3]-4\*y[n-2]+3\*y[n-1])/(2\*h)

    print("На основе полинома Лагранжа")

    print("Производная f("+str(t)+")="+str(round(y1,5)))

def difNewton(x,y,t): #производная на основе полинома Ньютона

    x,y = methodNewton(x,y) #интерполированные значения функции

    n = len(x) #количество значений

    h = (x[len(x)-1]-x[0])/n #шаг

    i = int((t-x[0])/h+h/2) #индекс точки близкой к t

    y1 = 0.0 #значение производной

    if i == 0:

        y1 = (-3\*y[0]+4\*y[1]-y[2])/(2\*h)

    elif i >0 and i < n:

        y1 = (-y[i-1]+y[i+1])/(2\*h)

    elif i == n:

        y1 = (y[n-3]-4\*y[n-2]+3\*y[n-1])/(2\*h)

    print("На основе полинома Ньютона")

    print("Производная f("+str(t)+")="+str(round(y1,5)))

#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 += 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

#Находит массив значений в точке t от x0 до xn

def methodNewton(x,y):

    t = x[0]

    xres = []

    yres = []

    while t < x[len(x)-1]:

        yres.append(newton(t,x,y))

        xres.append(t)

        t += 1

    return xres,yres

def newton(t,x,y):

    n = len(x)

    C = [] #создание матрицы

    for i in range(0,n):

        C.append([])

        C[i].append(y[i]) #заполнение первого столбца значениями y

        for j in range(0,n):

            C[i].append(0)

    for j in range(1,n): #получение значений матрицы

        for i in range(1,n):

            if i<j:

                C[i][j] = 0

            else:

                C[i][j] = (C[i][j-1]-C[j-1][j-1])/(x[i]-x[j-1])

    A = [] #диоганальные коэффициенты полученной матрицы

    for i in range(0,n):

        A.append(C[i][i])

    sum = A[n-1] #значение функции в точке t

    for i in range(n-2,-1,-1):

        sum = sum\*(t-x[i]) + A[i]

    return sum

main()