|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  **Федеральное государственное бюджетное образовательное учреждение**  **высшего образования**  **«Московский государственный технический университет**  **имени Н.Э. Баумана**  **(национальный исследовательский университет)»**  **(МГТУ им. Н.Э. Баумана)** |

ФАКУЛЬТЕТ **Информатика и системы управления**

КАФЕДРА **КОМПЬЮТЕРНЫЕ СИСТЕМЫ И СЕТИ (ИУ6)**

НАПРАВЛЕНИЕ ПОДГОТОВКИ **09.03.04 Программная инженерия**

**Отчет**

|  |
| --- |
| **по лабораторной работе № 6** |

Название:

**Построение и программная реализация алгоритмов численного дифференцирования.**

Дисциплина: Вычислительные алгоритмы

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Студент | ИУ7-46Б |  |  | Нгуен Ань Тхы |
|  | (Группа) |  | (Подпись, дата) | (И.О. Фамилия) |
|  |  |  |  |  |
| Преподаватель |  |  |  | Градов В.М. |
|  |  |  | (Подпись, дата) | (И.О. Фамилия) |

Москва, 2020

**Цель работы**: Получение навыков построения алгоритма вычисления производных от сеточных функций.

**I. Исходные данные.**

Задана табличная (сеточная) функция. Имеется информация, что закономерность, представленная этой таблицей, может быть описана формулой

,

параметры функции неизвестны **и определять их не нужно**..

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| x | y | 1 | 2 | 3 | 4 | 5 |
| 1 | 0.571 |  |  |  |  |  |
| 2 | 0.889 |  |  |  |  |  |
| 3 | 1.091 |  |  |  |  |  |
| 4 | 1.231 |  |  |  |  |  |
| 5 | 1.333 |  |  |  |  |  |
| 6 | 1.412 |  |  |  |  |  |

Вычислить первые разностные производные от функции и занести их в столбцы (1)-(4) таблицы:

1 - односторонняя разностная производная ,

2 - центральная разностная производная,

3- 2-я формула Рунге с использованием односторонней производной,

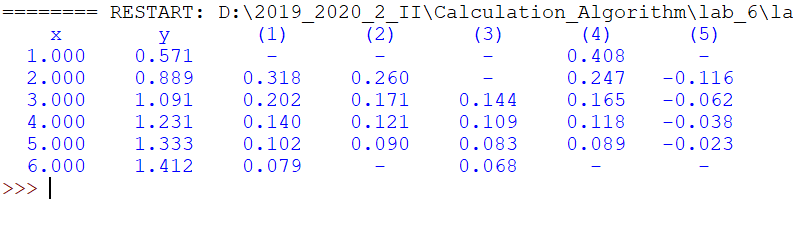
4 - введены выравнивающие переменные.

В столбец 5 занести вторую разностную производную.

**II. Результат работы программы.**

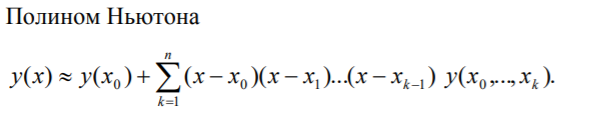
Заполненная таблица с краткими комментариями по поводу использованных формул и их точности

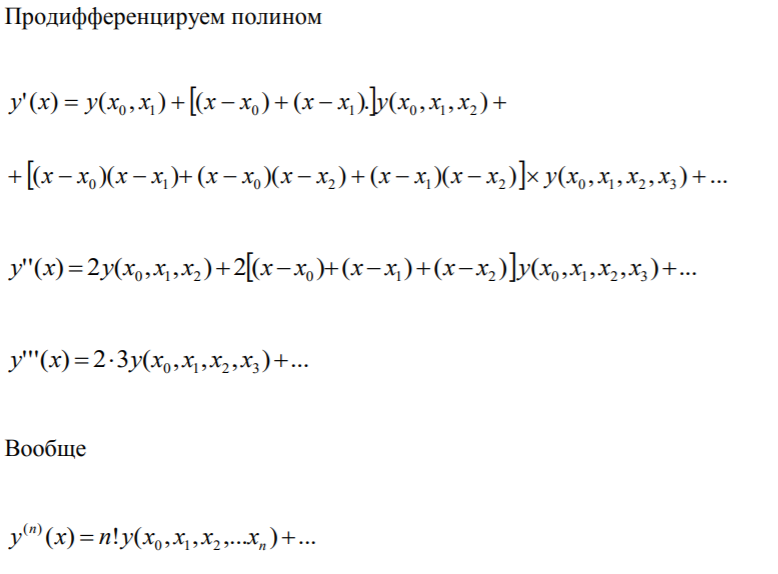
**III. Пример выполнения прграммы:**

****

**IV. Описание алгоритма:**

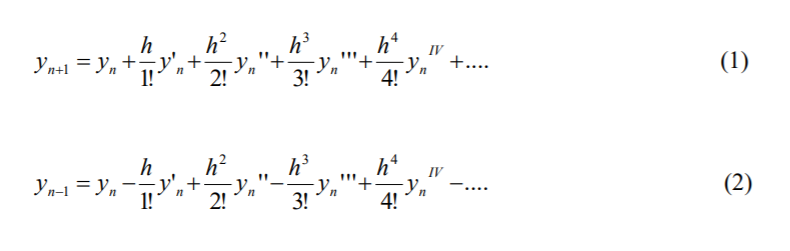
**1. Полиномиальные формулы**

****

****

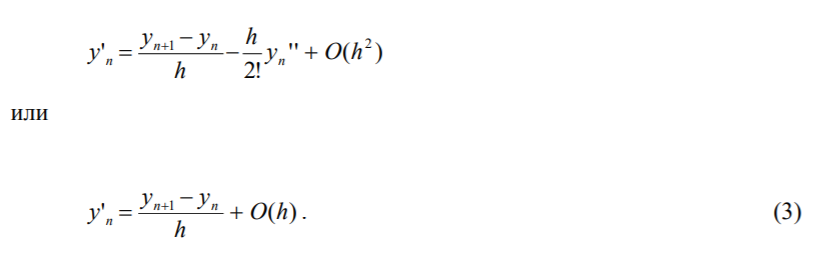
**2. Разложение в ряды Тейлора**

Выполним разложение функции в ряд Тейлора, приняв за центр разложения точку *xn*

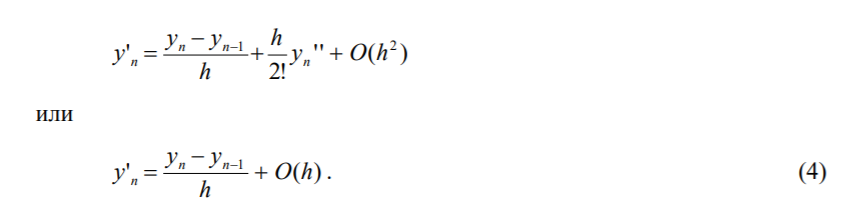


Получим разностные формулы для вычисления первых производных.

Из (1)

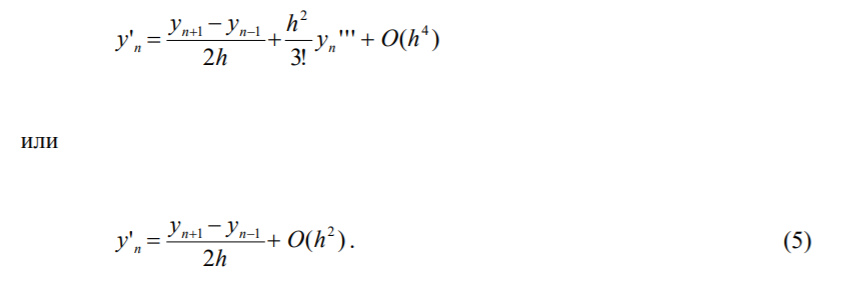


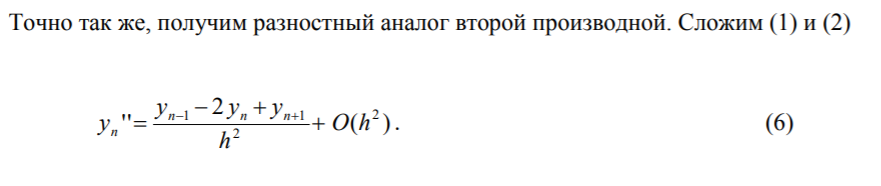
Из (2)

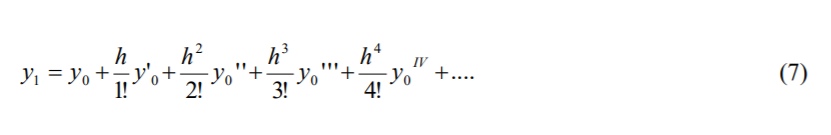


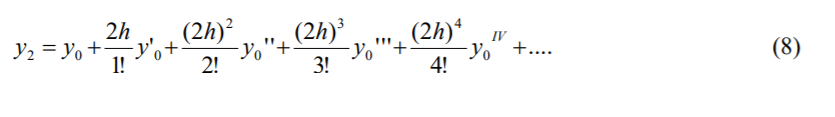
Выражение (3) называется **правой** разностной производной (или правосторонней формулой), (4) - **левой** разностной производной (или левосторонней формулой).

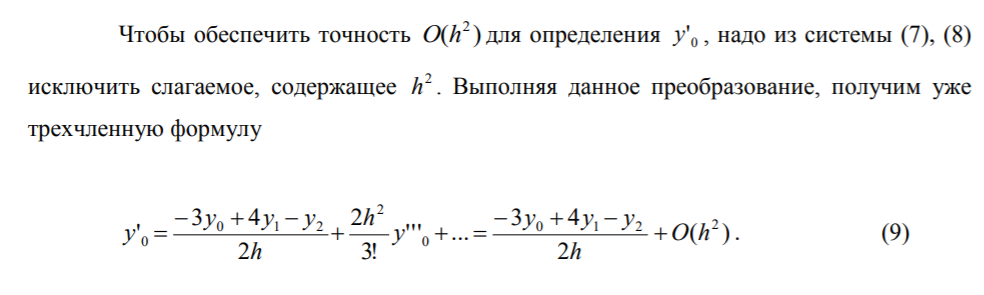
Вычитая (2) из (1), получим **центральную** формулу для первой производной

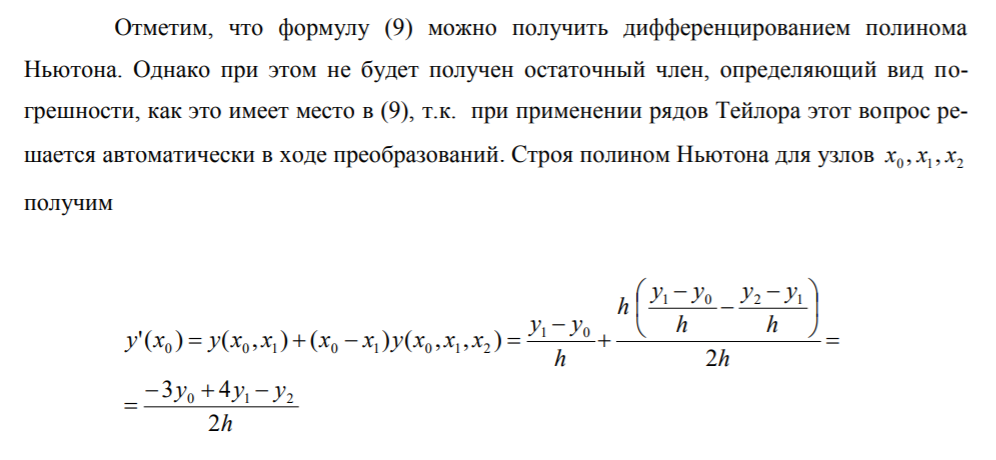




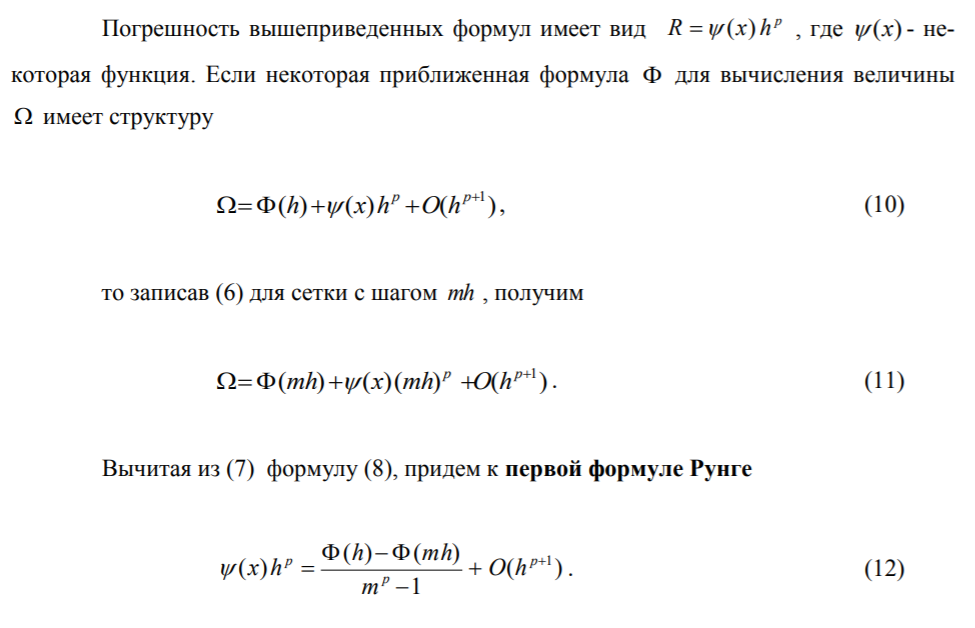


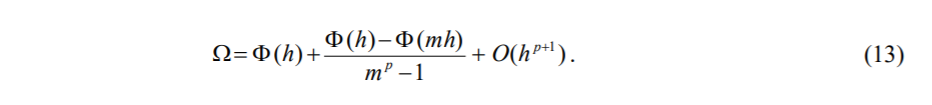


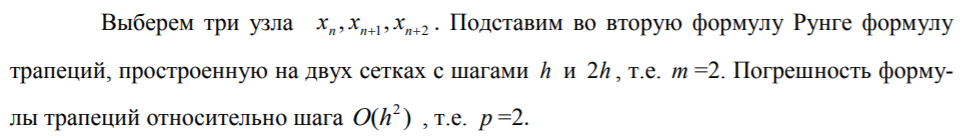


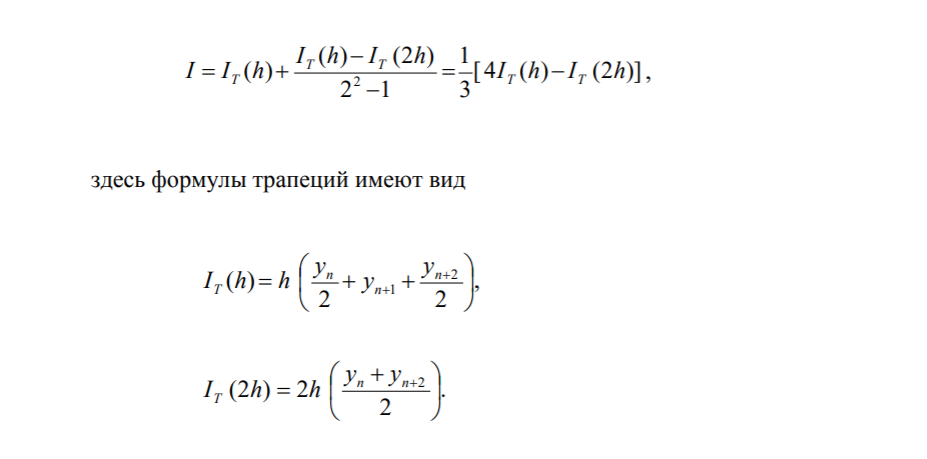


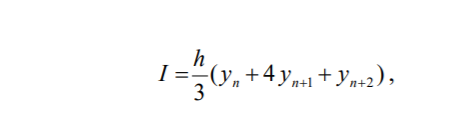
**3. Формулы Рунге**



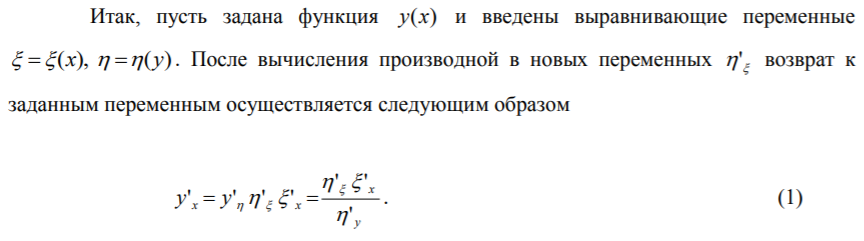




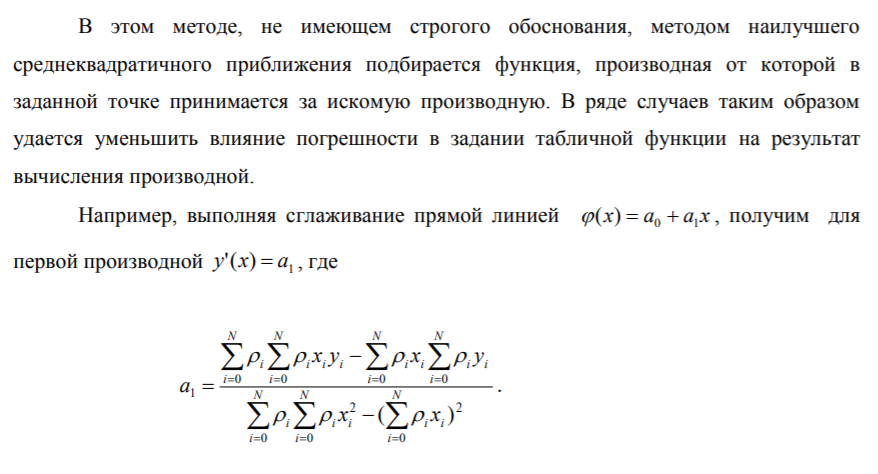




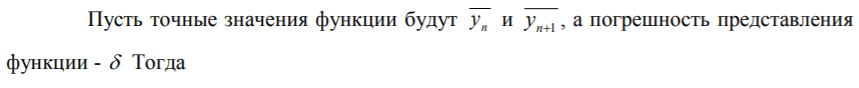
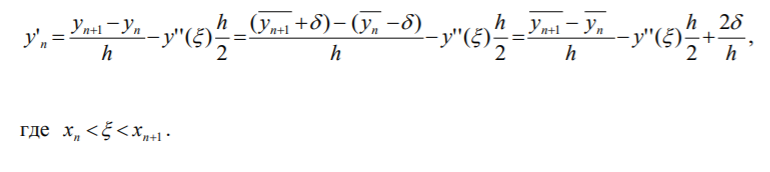
**4. Выравнивающие переменные**

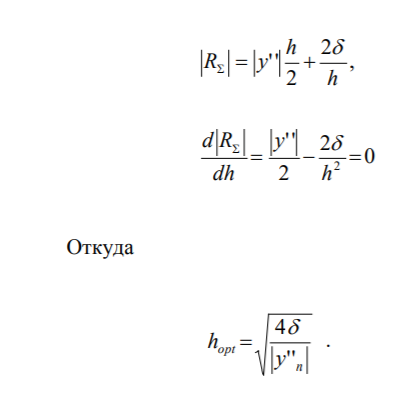
****

**5. Дифференцирование предварительно сглаженной кривой**

****

**6. Регуляризация дифференцирования**

**** ****

****

**V. Ответ на вопросы защита:**

1. Получить формулу порядка точности  для первой разностной производной  в крайнем правом узле .

2. Получить формулу порядка точности  для второй разностной производной  в крайнем левом узле .

3. Используя 2-ую формулу Рунге, дать вывод выражения (9) из Лекции №7 для первой производной  в левом крайнем узле

.

Формула Рунга при p = 1, m = 2

4. Любым способом из Лекций №7, 8 получить формулу порядка точности  для первой разностной производной  в крайнем левом узле .

**VI. Код программы:**

def LeftSide(y, h):

result = []

for i in range(len(y)):

if not i:

result.append("-")

else:

result.append(((y[i] - y[i - 1]) / h))

return result

def CenterDiff(y, h):

result = []

for i in range(len(y)):

if (not i) or (i == len(y) - 1):

result.append("-")

else:

result.append((y[i + 1] - y[i - 1]) / (2 \* h))

return result

def RungeLeft(y, h):

result = []

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

if i < 2:

result.append("-")

else:

result.append(2 \* ((y[i] - y[i - 1]) / h) - ((y[i] - y[i - 2]) / (2 \* h)))

return result

def Alignment(x, y, h):

result = []

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

if i > len(y) - 2:

result.append("-")

else:

result.append((1 / y[i + 1] - 1 / y[i]) / (1 / x[i + 1] - 1 / x[i]) \* (y[i] \*\* 2) / (x[i] \*\* 2))

return result

def SecondDiff(y, h):

result = []

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

if (not i) or (i > len(y) - 2):

result.append("-")

else:

result.append((y[i - 1] - 2 \* y[i] + y[i + 1]) / (h \*\* 2))

return result

def PrintResult(table):

print("{:^9}{:^9}{:^9}{:^9}{:^9}{:^9}{:^9}".format("x", "y", "(1)", "(2)", "(3)", "(4)", "(5)"))

N = len(table[1])

for i in range(N):

for result in table:

if result[i] == "-":

print("{:^9}".format(result[i]), end="")

else:

print("{:^9.3f}".format(result[i]), end="")

print()

def main():

h = 1

x = [i for i in range(1, 7)]

y = [0.571, 0.889, 1.091, 1.231, 1.333, 1.412]

left\_side = LeftSide(y, h)

center\_diff = CenterDiff(y, h)

runge\_left = RungeLeft(y, h)

alignment = Alignment(x, y, h)

second\_diff = SecondDiff(y, h)

table = [x, y, left\_side, center\_diff, runge\_left, alignment, second\_diff]

PrintResult(table)

main()