**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 «Программное обеспечение ЭВМ и информационные технологии»

**Лабораторная работа № 6**

**Тема** Построение и программная реализация алгоритмов численного

дифференцирования.

**Студент** Пронин А.С.

**Группа** ИУ7-42Б

**Оценка (баллы) \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_**

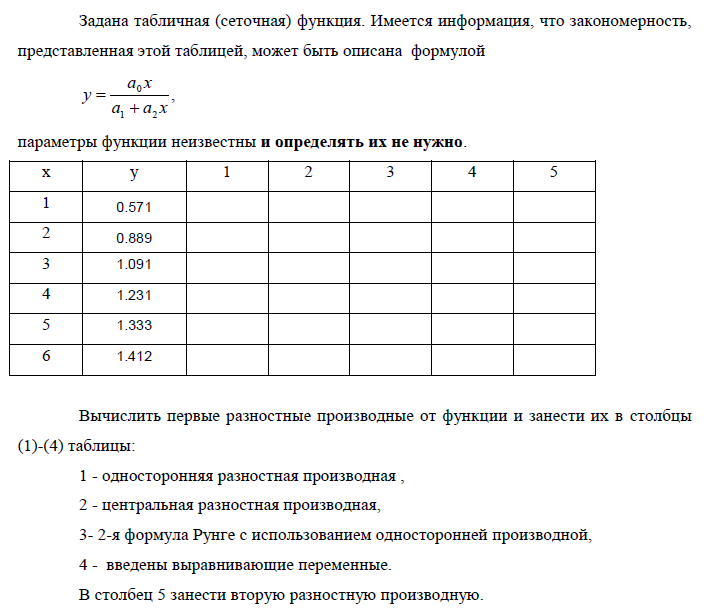
**Преподаватель** Градов В.М.

Москва.

2021 г

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

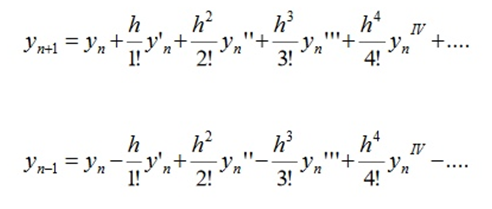
# ****1. Техническое задание****



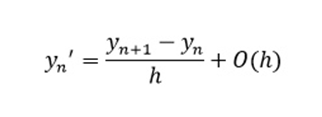
# 2. Алгоритм

## Односторонняя разностная производная

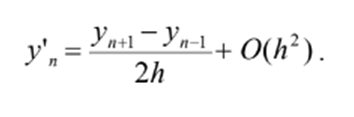
Выполним разложение функции в ряд Тейлора, приняв за центр разложения точку x(n):



Из первого разложения получим правую разностную производную:



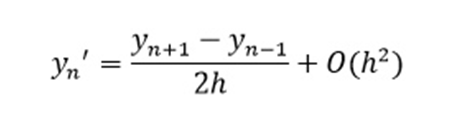
Из второго разложения получим левую разностную производную:



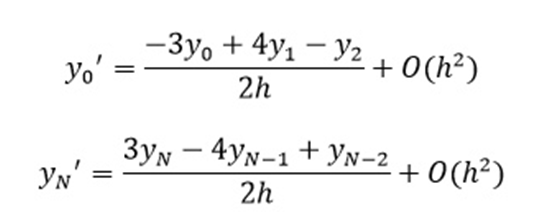
Эти формулы имеют наименьший, первый порядок точности относительно шага.

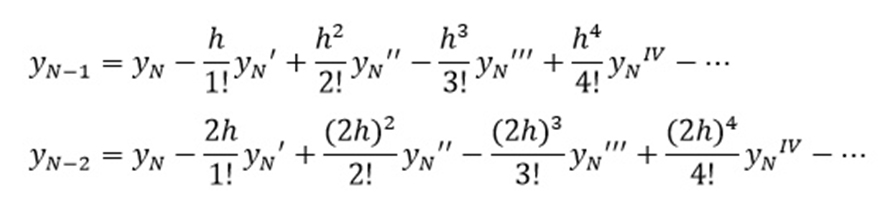
## Центральная разностная производная

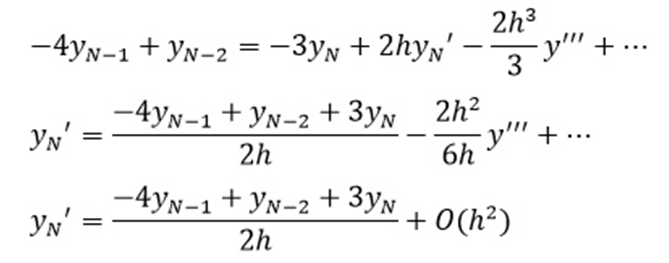
Из вышеописанного разложения по формуле Тейлора можно получить формулу для первой производной:



Для поиска производной в крайних узлах x0 и xN эта формула не подойдет. Формулы вычисления производной в крайних узлах:



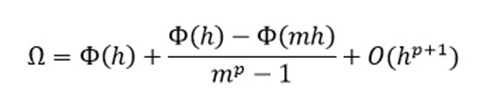




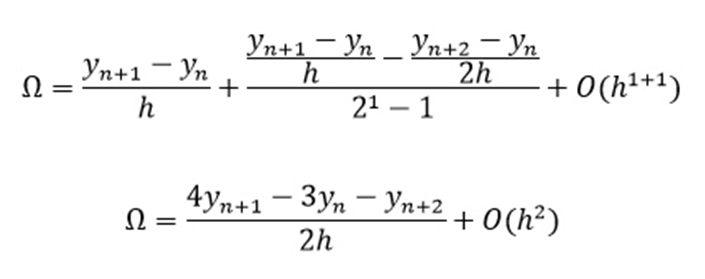
Выведенная формула имеет второй порядок точности.

## Вторая формула Рунге с использованием разностной производной.

Вторая формула Рунге:



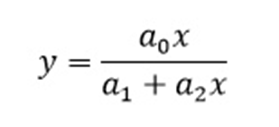
Подставим во вторую формулу Рунге правостороннюю формулу, построенную на двух сетках с шагами h и 2h, то есть m = 2. Погрешность правосторонней формулы относительно шага O(h), то есть p = 1.



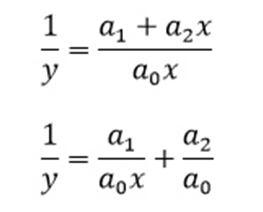
Выведенная формула имеет второй порядок точности.

## Метод выравнивающих переменных

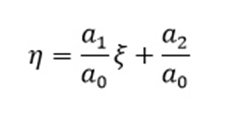
Известно, что табличная функция описывает некоторую закономерность вида:



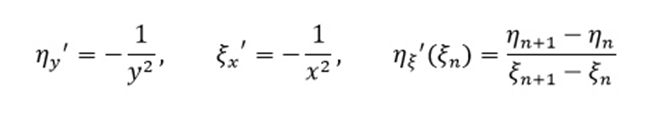
Введем выравнивающие переменные ξ=ξ(x), η=η(y). Для этого выполним следующие преобразования:



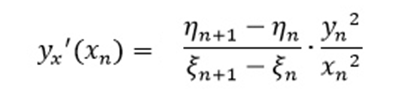
Пусть η=1/y и ξ=1/x , тогда:



Найдем следующие производные:



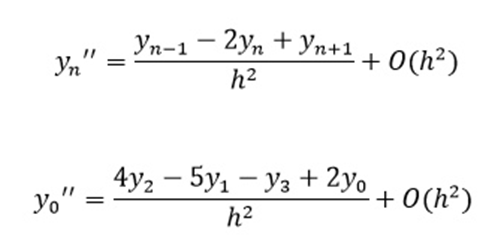
В результате получим формулу с использованием выравнивающих переменных:



Точность этого метода – 100%, без учёта аппаратных погрешностей.

## Вторая центральная производная

Формула выводится аналогично формуле для первой производной.

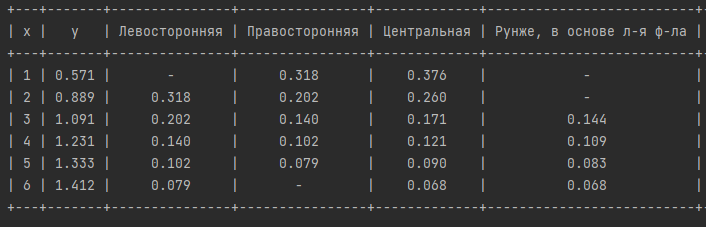


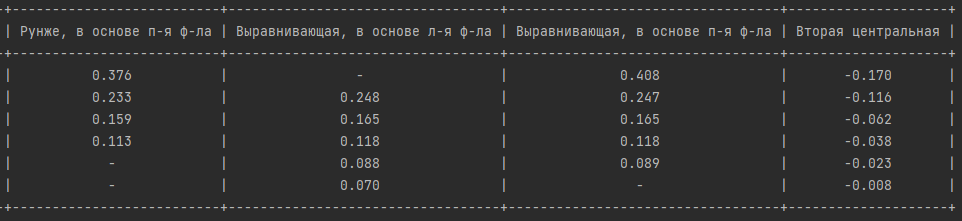
# 3. Код программы

from prettytable import PrettyTable  
  
# One sided  
# Правая разносторонняя производная  
def right(y\_cur, y\_next, step):  
 return (y\_next - y\_cur) / step  
  
# Левая разносторонняя производная  
def left(y\_cur, y\_prev, step):  
 return (y\_cur - y\_prev) / step  
  
def RightOneSided(ydata, step):  
 result = []  
 for i in range(len(ydata) - 1):  
 result.append("{:.3f}".format(right(ydata[i], ydata[i + 1], step)))  
 result.append("-")  
 return result  
  
def LeftOneSided(ydata, step):  
 result = ["-"]  
 for i in range(1, len(ydata)):  
 result.append("{:.3f}".format(left(ydata[i], ydata[i - 1], step)))  
 return result  
  
  
# Central  
# Центральная формула для левой производной  
def center(y\_next, y\_prev, step):  
 return (y\_next - y\_prev) / (2 \* step)  
  
# Центральная формула для x0  
def center\_x0(y\_0, y\_1, y\_2, step):  
 return (-3 \* y\_0 + 4 \* y\_1 - y\_2) / (2 \* step)  
  
# для xn  
def center\_xn(y\_n, yn\_1, yn\_2, step):  
 return (3 \* y\_n - 4 \* yn\_1 + yn\_2) / (2 \* step)  
  
def Central(input\_data, step):  
 result = [center\_x0(input\_data[0], input\_data[1], input\_data[2], step)]  
 length = len(input\_data)  
 for i in range(1, length - 1):  
 result.append(center(input\_data[i + 1], input\_data[i - 1], step))  
 result.append(center\_xn(input\_data[length - 1], input\_data[length - 2], input\_data[length - 3], step))  
 return ["{:.3f}".format(i) for i in result]  
  
  
#Runge  
# Вторая формула Рунге, в основе лежит правосторонняя формула  
def runge\_right(y\_cur, y\_next, y\_next\_next, step):  
 return (4 \* y\_next - 3 \* y\_cur - y\_next\_next) / (2 \* step)  
  
# Вторая формула Рунге, в основе лежит левосторонняя формула  
def runge\_left(y\_cur, y\_prev, y\_prev\_prev, step):  
 return (3 \* y\_cur - 4 \* y\_prev + y\_prev\_prev) / (2 \* step)  
  
def RightRunge(ydata, step):  
 result = []  
 for i in range(len(ydata) - 2):  
 result.append("{:.3f}".format(runge\_right(ydata[i], ydata[i + 1], ydata[i + 2], step)))  
 result.append("-")  
 result.append("-")  
 return result  
  
def LeftRunge(ydata, step):  
 result = ["-", "-"]  
 for i in range(2, len(ydata)):  
 result.append("{:.3f}".format(runge\_left(ydata[i], ydata[i - 1], ydata[i - 2], step)))  
 return result  
  
  
#Reshape  
def eta(y):  
 return 1 / y  
  
def ksi(x):  
 return 1 / x  
  
# В основе лежит правосторонняя формула  
def right\_reshape(y\_cur, y\_next, x\_cur, x\_next):  
 return (eta(y\_next) - eta(y\_cur)) / (ksi(x\_next) - ksi(x\_cur)) \* (y\_cur / x\_cur) \*\* 2  
  
# В основе лежит левосторонняя формула.  
def left\_reshape(y\_cur, y\_prev, x\_cur, x\_prev):  
 return (eta(y\_cur) - eta(y\_prev)) / (ksi(x\_cur) - ksi(x\_prev)) \* (y\_cur / x\_cur) \*\* 2  
  
def RightReshape(xdata, ydata):  
 result = []  
 for i in range(len(ydata) - 1):  
 result.append("{:.3f}".format(right\_reshape(ydata[i], ydata[i + 1], xdata[i], xdata[i + 1])))  
 result.append("-")  
 return result  
  
def LeftReshape(xdata, ydata):  
 result = ["-"]  
 for i in range(1, len(ydata)):  
 result.append("{:.3f}".format(left\_reshape(ydata[i], ydata[i - 1], xdata[i], xdata[i - 1])))  
 return result  
  
  
#Second central  
def second\_central(y\_cur, y\_prev, y\_next, step):  
 return (y\_prev - 2 \* y\_cur + y\_next) / step \*\* 2  
  
def second\_central\_x0(y0, y1, y2, y3, step):  
 return (4 \* y2 - 5 \* y1 - y3 + 2 \* y0) / step \*\* 2  
  
def second\_central\_xn(yn, yn\_1, yn\_2, yn\_3, step):  
 return (4 \* yn\_2 - 5 \* yn\_1 - yn\_3 + 2 \* yn) / step \*\* 2  
  
def SecondCentral(ydata, step):  
 result = ["{:.3f}".format(second\_central\_x0(ydata[0], ydata[1], ydata[2], ydata[3], step))]  
 for i in range(1, len(ydata) - 1):  
 result.append("{:.3f}".format(second\_central(ydata[i], ydata[i - 1], ydata[i + 1], step)))  
 n = len(ydata) - 1  
 result.append("{:.3f}".format(second\_central\_xn(ydata[n], ydata[n - 1], ydata[n - 2], ydata[n - 3], step)))  
 return result

#Main  
def main():  
 xdata = [1, 2, 3, 4, 5, 6]  
 ydata = [0.571, 0.889, 1.091, 1.231, 1.333, 1.412]  
  
 table = [xdata, ydata, [], [], [], [], []]  
  
 step = 1  
  
 data\_12 = RightOneSided(ydata, step)  
 data\_11 = LeftOneSided(ydata, step)  
 data\_2 = Central(ydata, step)  
 data\_32 = RightRunge(ydata, step)  
 data\_31 = LeftRunge(ydata, step)  
 data\_42 = RightReshape(xdata, ydata)  
 data\_41 = LeftReshape(xdata, ydata)  
 data\_5 = SecondCentral(ydata, step)  
  
 table = PrettyTable()  
 table.add\_column("x", xdata)  
 table.add\_column("y", ydata)  
 table.add\_column("Левосторонняя", data\_11)  
 table.add\_column("Правосторонняя", data\_12)  
 table.add\_column("Центральная", data\_2)  
 table.add\_column("Рунже, в основе л-я ф-ла", data\_31)  
 table.add\_column("Рунже, в основе п-я ф-ла", data\_32)  
 table.add\_column("Выравнивающая, в основе л-я ф-ла", data\_41)  
 table.add\_column("Выравнивающая, в основе п-я ф-ла", data\_42)  
 table.add\_column("Вторая центральная", data\_5)  
  
 print(table)  
  
if \_\_name\_\_ == '\_\_main\_\_':  
 main()

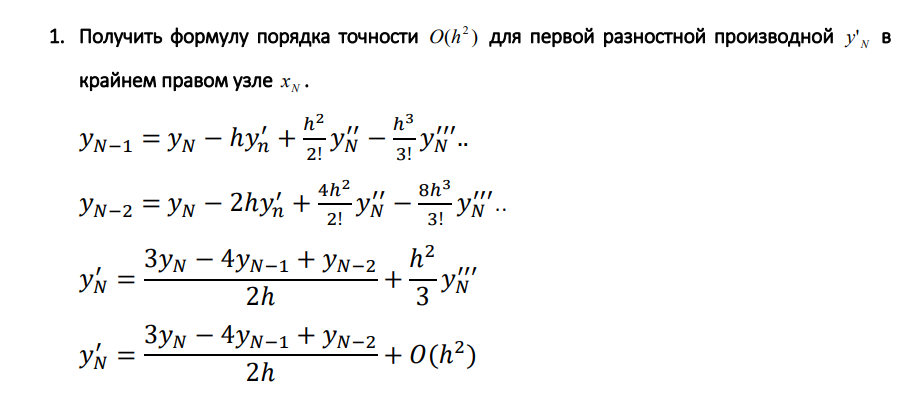
# 5. Результаты работы





# 6. Вопросы при защите лабораторной работы

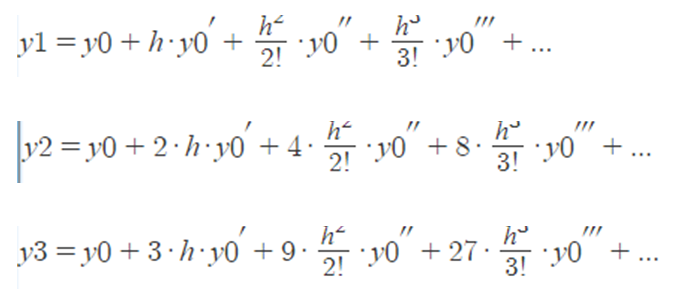
***1. Получить формулу порядка точности для первой разностной производной в крайнем правом узле .***

******

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

С использованием рядов Тейлора:

Выполним разложение в ряд Тейлора:



Теперь цель избавиться от h, h^3

Для этого умножаем первое уравнение на -2 и складываем со вторым.

Получим:



Упростим:



Далее умножаем первое уравнение на -3 и складываем с третьим.

Получим:



Упростим:

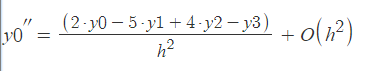


На данном шаге умножаем четвёртое уравнение на -4, складываем с пятым и приводим подобные слагаемые.

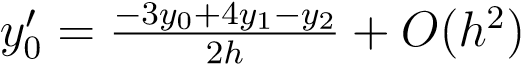
Получим:



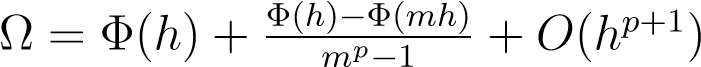
Откуда:



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



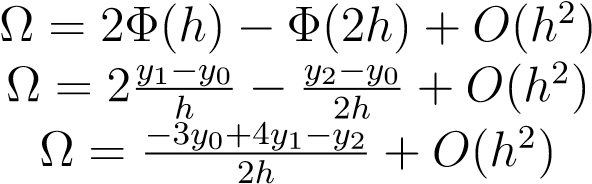
По формуле Рунге:



p = 1, т.к. исходная формула имеет первый порядок точности

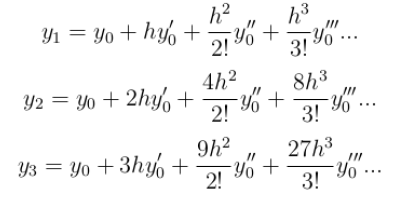
m = 2 для удобства

В итоге:



***4. Любым способом из Лекций №7, 8 получить формулу порядка точности***

***для первой разностной производной в крайнем левом узле .***



Откуда получим:

