|  |  |
| --- | --- |
| 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 **Программное обеспечение ЭВМ и информационные технологии**

**Лабораторная работа №6.**

**«Построение и программная реализация алгоритмов численного дифференцирования»**

Студент **Леонов Владислав Вячеславович**

Группа **ИУ7-46Б**

Студент **\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_** Леонов В.В.

*подпись, дата фамилия, и.о.*

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ Градов В.М.

*подпись, дата фамилия, и.о.*

Оценка \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

*Москва, 2021 г.*

**Оглавление**

[Цель работы 3](#_Toc69758330)

[Исходные данные 3](#_Toc69758331)

[Описание алгоритма 3](#_Toc69758332)

[Код программы 6](#_Toc69758333)

[Результаты работы 9](#_Toc69758334)

[Ответы на контрольные вопросы 11](#_Toc69758335)

# Цель работы

Получение навыков построения алгоритма вычисления производных от сеточных функций.

# Исходные данные

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

Параметры функции неизвестны и определять их не нужно.

Вычислить значения производных и занести их в таблицу:

* 1 столбец – односторонняя разностная производная.
* 2 столбец – центральная разностная производная.
* 3 столбец – вторая формула Рунге с односторонней производной.
* 4 столбец – введены выравнивающие переменные.
* 5 столбец – вторая разностная производная.

# Описание алгоритма

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

Получаем формулы для вычисления первых производных:

или

Первое выражение – **правая разностная производная**, второе – **левая разностная производная**. Эти формулы имеют самый низкий, первый порядок точности.

Аналогичным образом, можно получить **центральную формулу** для первой производной:

Данная формула имеет более высокий, второй порядок точности. Точно так же, получим **разностный аналог второй производной**:

Из разности двух рядов Тейлора можно вывести **первую формулу Рунге:**

Комбинируя следующие формулы получим вторую формулу Рунге:

Заметим, что вывод формулы Рунге мы провели на примере операции дифференцирования. Однако все выкладки справедливы для любых других приближенных вычислений. Важно только, чтобы погрешность применяемых формул имела вид .

Идея **введения выравнивающих переменных**, рассмотренная при изучении аппроксимации функций, очень хорошо работает при проведении операций дифференцирования. Действительно, при удачном выборе этих переменных исходная кривая может быть преобразована в прямую линию, производная от которой вычисляется точно по самым простым формулам. Итак, пусть задана функция и введены выравнивающие переменные . После вычисления производной в новых переменных возврат к предыдущим переменным осуществляется следующим образом:

В новых переменных значение производной можно вычислить по любой односторонней формуле.

# Код программы

Код программы представлен ниже.

|  |
| --- |
| Файл ***main.py*** |
| **from** config **import** DATA\_FILE  **from** data **import** data\_read, data\_print  **from** differ\_methods **import** differ\_methods      **def** main():  data = data\_read(DATA\_FILE)  data\_print(data)  differ\_methods(data)      **if** \_\_name\_\_ == "\_\_main\_\_":  main() |

|  |
| --- |
| Файл ***config.py*** |
| DATA\_FILE = "data.txt" |

|  |
| --- |
| Файл ***data.py*** |
| **from** dataclasses **import** dataclass      @dataclass  **class** FunctionData:  X: list[float]  Y: list[float]    **def** \_\_init\_\_(self):  self.X = []  self.Y = []    **def** append(self, x: float, y: float):  self.X.append(x)  self.Y.append(y)    **def** data\_read(filename: str) -> FunctionData:  data = FunctionData()    f = open(filename, "r")  **for** line **in** f:  line = (line.replace("**\n**", "")).split(" ")  x, y = [float(num) **for** num **in** line]  data.append(x, y)  f.close()    **return** data      **def** data\_print(data: FunctionData) -> None:  **for** (x, y) **in** zip(data.X, data.Y):  **print**("X = {:.3f}, Y = {:.3f}".format(x, y)) |

|  |
| --- |
| Файл ***differ\_methods.py*** |
| **from** data **import** FunctionData      **def** f\_der\_value(y1: float, y2: float, st: float) -> float:  **return** (y2 - y1) / st      **def** s\_der\_value(y1: float, y2: float, y3: float, st: float) -> float:  **return** (y1 - 2 \* y2 + y3) / st      **class** Derivative():  **def** \_\_init\_\_(self):  self.values: list[float] = []    **def** clear(self):  self.values.clear()    **def** prnt(self):  **for** value **in** self.values:  **print**("{:10.5f}".format(value), end=' ')  **print**()    **def** left(self, data: FunctionData):  st = data.X[1] - data.X[0]  **for** i **in** range(len(data.Y)):  **if** i > 0:  value = f\_der\_value(data.Y[i - 1], data.Y[i], st)  **else**:  value = float('NaN')  self.values.append(value)    **def** right(self, data: FunctionData):  st = data.X[1] - data.X[0]  **for** i **in** range(len(data.Y)):  **if** i < len(data.Y) - 1:  value = f\_der\_value(data.Y[i], data.Y[i + 1], st)  **else**:  value = float('NaN')  self.values.append(value)      **def** center(self, data: FunctionData):  st = 2 \* (data.X[1] - data.X[0])  **for** i **in** range(len(data.Y)):  **if** i > 0 **and** i < len(data.Y) - 1:  value = f\_der\_value(data.Y[i - 1], data.Y[i + 1], st)  **else**:  value = float('NaN')  self.values.append(value)    **def** runge(self, data: FunctionData):  st = (data.X[1] - data.X[0])  **for** i **in** range(len(data.Y)):  **if** i > 1:  d1 = f\_der\_value(data.Y[i - 1], data.Y[i], st)  d2 = f\_der\_value(data.Y[i - 2], data.Y[i], 2 \* st)  value = 2 \* d1 - d2  **else**:  value = float('NaN')  self.values.append(value)    **def** align\_var(self, data: FunctionData):  **for** i **in** range(len(data.Y)):  **if** i < len(data.Y) - 1:  eta\_ksi\_diff = (1 / data.Y[i + 1] - 1 / data.Y[i]) / (  1 / data.X[i + 1] - 1 / data.X[i])  value = eta\_ksi\_diff \* data.Y[i]\*\*2 / data.X[i]\*\*2  **else**:  value = float('NaN')  self.values.append(value)    **def** scnd\_diff(self, data: FunctionData):  st = (data.X[1] - data.X[0])\*\*2  **for** i **in** range(len(data.Y)):  **if** i > 0 **and** i < len(data.Y) - 1:  value = s\_der\_value(data.Y[i - 1], data.Y[i], data.Y[i + 1],st)  **else**:  value = float('NaN')  self.values.append(value)      **def** differ\_methods(data: FunctionData) -> None:  deriv = Derivative()  methods = [  deriv.left, deriv.right, deriv.center, deriv.runge, deriv.align\_var,  deriv.scnd\_diff  ]    **for** method **in** methods:  **print**("{:15}".format(method.\_\_name\_\_), end=" ")  method(data)  deriv.prnt()  deriv.clear() |

# Результаты работы

В результате работы программы получаем таблицу производных:

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| X | Y | 1 | 2 | 3 | 4 | 5 |
| 1 | 0.571 | - | - | - | 0.409 | - |
| 2 | 0.889 | 0.318 | 0.260 | - | 0.247 | -0.116 |
| 3 | 1.091 | 0.202 | 0.171 | 0.144 | 0.166 | -0.062 |
| 4 | 1.231 | 0.140 | 0.121 | 0.109 | 0.118 | -0.038 |
| 5 | 1.333 | 0.102 | 0.091 | 0.083 | 0.090 | -0.023 |
| 6 | 1.412 | 0.079 | - | 0.068 | - | - |

***Первый столбец:***

Используем левостороннюю формулу, в точке производная не определена. Фактическая точность совпадает с ожидаемой .

***Второй столбец:***

Используем центральную формулу, в точках и производная не определена. Можно заметить, что точность .

***Третий столбец:***

Используем вторую формулу Рунге на основе левой разностной производной. Т.к. формула Рунге повышает точность на один порядок, а левая разностная формула обеспечивает первый порядок точности, то в итоге получаем второй порядок точности . Это можно подтвердить тем, что результат довольно близок по значению к центральной формуле.

***Четвертый столбец:***

Используем выравнивающие переменные. Исходя из того, что значения производной близки по значению к производным, вычисленным через центральную формулу и формулу Рунге, можно предположить в данном конкретном случае данный метод схожую точность .

***Пятый столбец:***

Используем разностную формулу второй производной. Данная формула обеспечивает второй порядок точности

# Ответы на контрольные вопросы

***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 вторую формулу Рунге, дать вывод выражения (7) из лекции №7 для первой производной в левом узле.***

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