**Министерство науки и высшего образования Российской Федерации**

Федеральное государственное автономное образовательное учреждение высшего образования «Национальный исследовательский университет ИТМО»

Факультет программной инженерии и компьютерной техники

**Дисциплина:** Вычислительная математика

**Лабораторная работа №6**

**«Численное дифференцирование»**

**Вариант №10**

Выполнил:

Мокров Семён Андреевич

Группа:

P3215

Проверила:

Малышева Татьяна Алексеевна

Санкт-Петербург

2022

# Цель работы

Решить задачу Коши численными методами.

Для исследования использовать:

* Одношаговые методы;
* Многошаговые методы.

# Задание лабораторной работы

1. Программная реализация задачи:

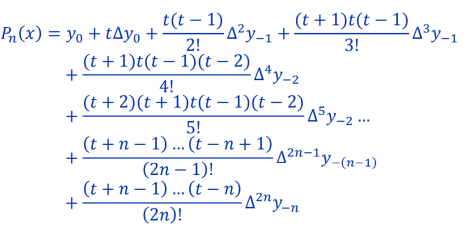
* Исходные данные: ОДУ вида , начальные условия , интервал дифференцирования [*a, b*], шаг *h*, точность .
* Составить таблицу приближенных значений интеграла дифференциального уравнения, удовлетворяющего начальным условиям. Для оценки точности использовать правило Рунге.
* Построить графики точного решения и полученного численного решения (разными цветами).

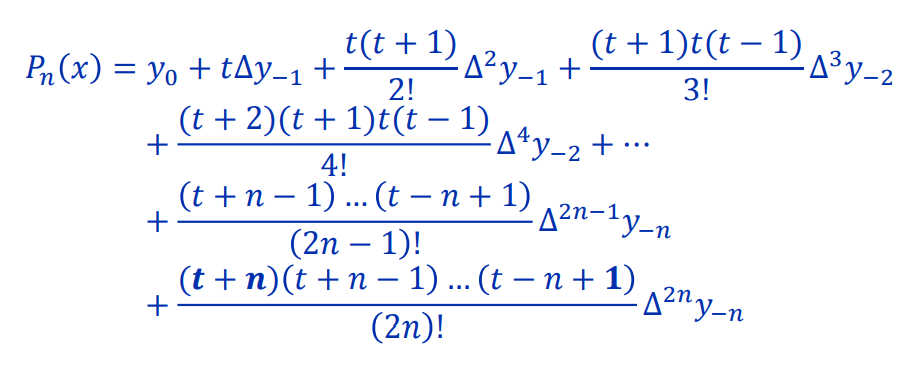
# Рабочие формулы

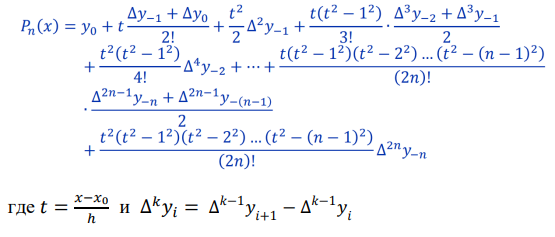
* Формула для полинома Лагранжа:



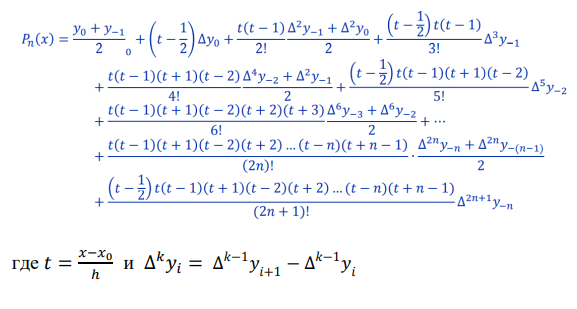
* Формула интерполяционного полинома Гаусса для x > a:

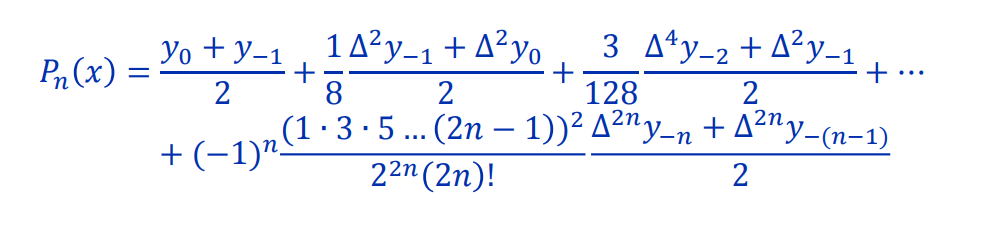


* Формула интерполяционного полинома Гаусса для x < a:
* Формула интерполяционного полинома Стирлинга:



* Формула интерполяционного полинома Бесселя:



***При t = 0.5:***

# Вычислительная реализация задачи

**Метод Гаусса:**

|  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- |
| i | x | y |  |  |  |  |  |  |
| -3 | 0,50 | 1,5320 | 1,0036 | 0,0014 | -0,0008 | -0,001 | 0,0059 | -0,017 |
| -2 | 0,55 | 2,5356 | 1,005 | 0,0006 | -0,002 | 0,0047 | -0,011 | - |
| -1 | 0,60 | 3,5406 | 1,0056 | -0,001 | 0,0027 | -0,006 | - | - |
| 0 | 0,65 | 4,5462 | 1,0042 | 0,0013 | -0,0033 | - | - | - |
| 1 | 0,70 | 5,5504 | 1,0055 | -0,002 | - | - | - | - |
| 2 | 0,75 | 6,5559 | 1,0035 | - | - | - | - | - |
| 3 | 0,80 | 7,5594 | - | - | - | - | - | - |

* Для ***x = 0,523***:

* Для ***x = 0,639***:

**Метод Ньютона:**

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| x | y |  |  |  |  |  |  |
| 0,50 | 1,5320 | 1,0036 | 0,0014 | -0,0008 | -0,001 | 0,0059 | -0,017 |
| 0,55 | 2,5356 | 1,005 | 0,0006 | -0,002 | 0,0047 | -0,011 | - |
| 0,60 | 3,5406 | 1,0056 | -0,001 | 0,0027 | -0,006 | - | - |
| 0,65 | 4,5462 | 1,0042 | 0,0013 | -0,0033 | - | - | - |
| 0,70 | 5,5504 | 1,0055 | -0,002 | - | - | - | - |
| 0,75 | 6,5559 | 1,0035 | - | - | - | - | - |
| 0,80 | 7,5594 | - | - | - | - | - | - |

* Для ***x = 0,523***:

*x`*

* Для ***x = 0,639***:

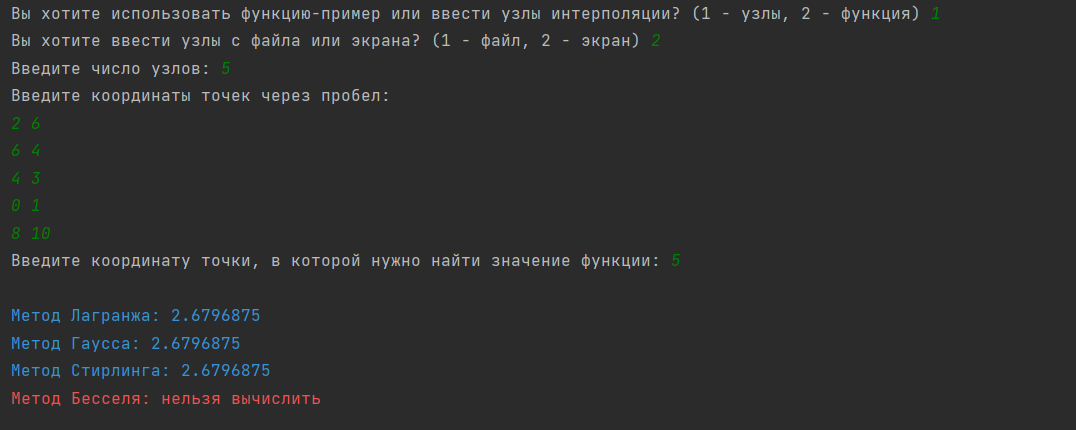
# Листинг программы

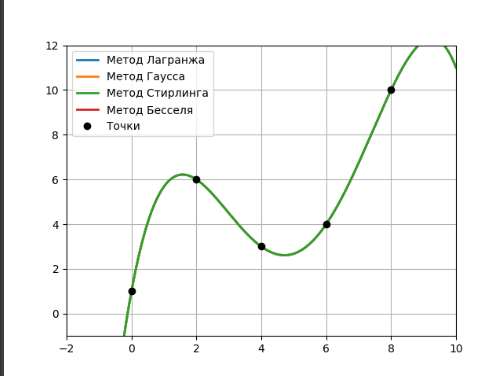
Весь исходный код программы располагается на GitHub (адрес: https://github.com/semwett0301/lab5\_math)

import math  
  
  
def lagrange\_method(points, x):  
 result = 0  
 for i in range(len(points)):  
 num = 1  
 den = 1  
 for j in range(len(points)):  
 if i != j:  
 num = num \* (x - points[j][0])  
 den = den \* (points[i][0] - points[j][0])  
 result = result + points[i][1] \* num / den  
 return result  
  
  
def check\_points(points, n):  
 h = points[1][0] - points[0][0]  
 for i in range(n - 1):  
 if round(points[i + 1][0] - points[i][0], 8) != round(h, 8):  
 return  
 return h  
  
  
def gauss\_method(points, x):  
 def calculate\_t\_first(t, length):  
 tmp = t  
 k = 1  
 for i in range(1, length):  
 if i % 2 == 1:  
 tmp \*= (t - k)  
 else:  
 tmp \*= (t + k)  
 k += 1  
 return tmp  
  
 def calculate\_t\_second(t, length):  
 tmp = t  
 k = 1  
 for i in range(1, length):  
 if i % 2 == 1:  
 tmp \*= (t + k)  
 else:  
 tmp \*= (t - k)  
 k += 1  
 return tmp  
  
 n = len(points)  
 h = check\_points(points, n)  
 if h is None:  
 return  
  
 y = [[0 for i in range(n)] for j in range(n)]  
  
 for i in range(n):  
 y[i][0] = points[i][1]  
  
 for i in range(1, n):  
 for j in range(n - i):  
 y[j][i] = y[j + 1][i - 1] - y[j][i - 1]  
  
 if n % 2 == 1:  
 a = n // 2  
 else:  
 a = n // 2 - 1  
  
 result = y[a][0]  
 t = (x - points[a][0]) / h  
 if x > points[a][0]:  
 for i in range(1, n):  
 if n % 2 == 1:  
 result += calculate\_t\_first(t, i) \* y[int((n - i) / 2)][i] / math.factorial(i)  
 else:  
 result += calculate\_t\_first(t, i) \* y[int((n - i - 1) / 2)][i] / math.factorial(i)  
 elif x < points[a][0]:  
 for i in range(1, n):  
 if n % 2 == 1:  
 result += calculate\_t\_second(t, i) \* y[int((n - i - 1) / 2)][i] / math.factorial(i)  
 else:  
 result += calculate\_t\_second(t, i) \* y[int((n - i - 2) / 2)][i] / math.factorial(i)  
  
 return result  
  
  
def get\_diff(y):  
 fin\_diff = [y]  
 for i in range(len(y) - 1):  
 temp\_fin\_dif = []  
 for j in range(len(fin\_diff[i]) - 1):  
 diff = fin\_diff[i][j + 1] - fin\_diff[i][j]  
 temp\_fin\_dif.append(diff)  
 fin\_diff.append(temp\_fin\_dif)  
 return fin\_diff  
  
  
def stirling\_method(points, arg):  
 n = len(points)  
 if n % 2 == 0:  
 return  
  
 h = check\_points(points, n)  
 if h is None:  
 return  
  
 y = [i[1] for i in points]  
  
 fin\_diff = get\_diff(y)  
 middle = len(points) // 2  
 t = (arg - points[middle][0]) / h  
 result = points[middle][1]  
  
 for step in range(1, middle + 1):  
 mul = 1  
 for j in range(1, step):  
 mul \*= (t \* t - j \* j)  
 result += 1 / math.factorial(2 \* step - 1) \* t \* mul \* (  
 fin\_diff[2 \* step - 1][-(step - 1) + middle] + fin\_diff[2 \* step - 1][-step + middle]) / 2  
 result += 1 / math.factorial(2 \* step) \* (t \*\* 2) \* mul \* (fin\_diff[2 \* step][-step + middle])  
 return result  
  
  
def bessel\_method(points, arg):  
 n = len(points)  
 if n % 2 == 1:  
 return  
  
 h = check\_points(points, n)  
 if h is None:  
 return  
  
 y = [i[1] for i in points]  
  
 fin\_diff = get\_diff(y)  
 middle = (len(points) - 2) // 2  
 t = (arg - points[middle][0]) / h  
 result = 0  
  
 for step in range(0, middle + 1):  
 if t != 0.5:  
 mul = 1  
 for j in range(1, step + 1):  
 mul \*= (t - j) \* (t + j - 1)  
 result += (1 / math.factorial(2 \* step)) \* mul \* (fin\_diff[2 \* step][-step + middle] + fin\_diff[2 \* step][-(step - 1) + middle]) / 2  
 result += (1 / math.factorial(2 \* step + 1)) \* (t - (1 / 2)) \* mul \* (fin\_diff[2 \* step + 1][-step + middle])  
 else:  
 mul = 1  
 k = 1  
 for j in range(1, step + 1):  
 mul \*= k  
 k += 2  
 result += (-1) \*\* step / 2 \* mul \*\* 2 / (2 \*\* (2 \* step)) / math.factorial(2 \* step) \  
 \* (fin\_diff[2 \* step][-step + middle] + fin\_diff[2 \* step][-(step - 1) + middle])  
 return result  
  
  
methods = [  
 [lagrange\_method, "Метод Лагранжа"],  
 [gauss\_method, "Метод Гаусса"],  
 [stirling\_method, "Метод Стирлинга"],  
 [bessel\_method, "Метод Бесселя"]  
]

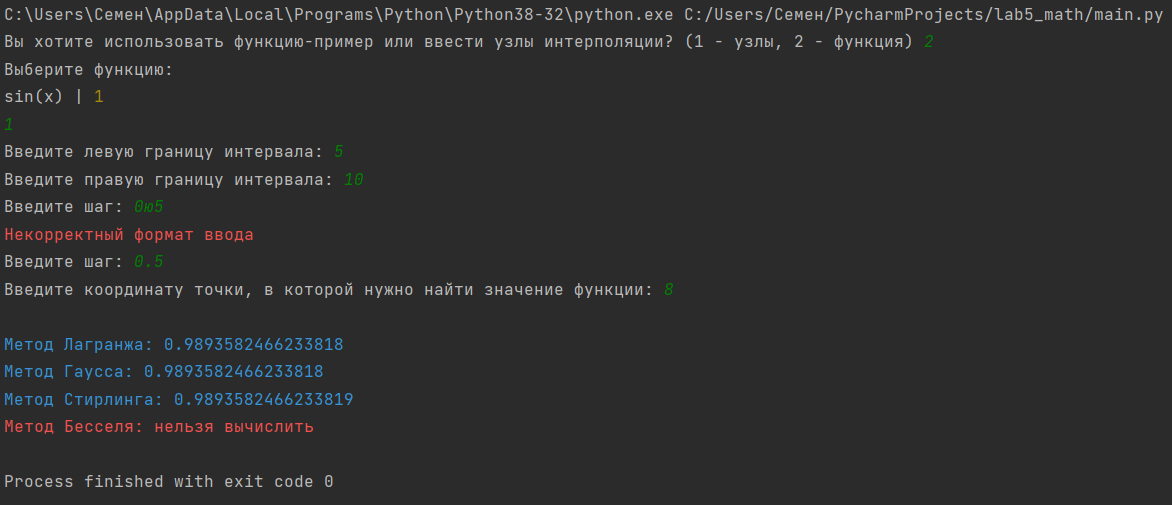
# Примеры и результаты работы программы

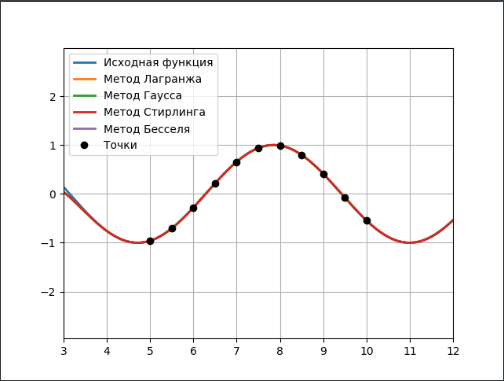
**При вводе точек:**

****

****

**При использовании функции:**

****

****

# Выводы

В данной лабораторной работе я:

* Познакомился с принципами интерполяции функций;
* Ознакомился с методами интерполяции Лагранжа, Гаусса, Стирлинга и Бесселя;
* Реализовал их на языке Python;
* Все методы выдают одинаковый результат с малой погрешностью (она связана с вычислительной погрешностью).