БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ

Отчёт по теме

«Аппроксимация и интерполяция функций»

Вариант 9

Выполнили: студенты гр. 853503

Климович А. А.

Галиева Э. Г. Ю.

Ивойлов О. А.

Проверил:

Протько М. И.

Минск 2020

**Содержание**

[Введение 3](#_Toc41567113)

[Конечная разность 3](#_Toc41567114)

[Разделённые разности 3](#_Toc41567115)

[Интерполяционный метод Лагранжа 5](#_Toc41567116)

[Интерполяционная формула Ньютона 5](#_Toc41567117)

[Интерполяция сплайнами 6](#_Toc41567118)

[Листинги программ и результаты 7](#_Toc41567119)

[Задание 1 7](#_Toc41567120)

[Код программы и результаты к заданию 1 7](#_Toc41567121)

[Результаты работы 11](#_Toc41567122)

[Задание 2 15](#_Toc41567123)

[Код программы и результаты к заданию 2: 15](#_Toc41567124)

[Результаты работы: 16](#_Toc41567125)

[Вывод 17](#_Toc41567126)

# Введение

## Конечная разность

**.**

Конечная разность — [математический](https://ru.wikipedia.org/wiki/%D0%9C%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) термин, широко применяющийся в [методах вычисления](https://ru.wikipedia.org/wiki/%D0%92%D1%8B%D1%87%D0%B8%D1%81%D0%BB%D0%B8%D1%82%D0%B5%D0%BB%D1%8C%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) при [интерполировании](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D1%8F).

Рассмотрим интерполяционную задачу для функции :

, ..., ,

где , .

**Конечной разностью 1-го порядка** называют разность между двумя соседними значениями в узлах интерполяции, то есть

, .

**Конечной разностью 2-го порядка** называют разность между двумя соседними конечными разностями 1-го порядка, то есть

, .

**Конечной разностью порядка**  (для ) называют разность между двумя соседними конечными разностями порядка то есть

.

Если ввести оператор смещения такой, что , то оператор восходящей конечной разности и

,

который можно раскладывать по [биному Ньютона](https://ru.wikipedia.org/wiki/%D0%91%D0%B8%D0%BD%D0%BE%D0%BC_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0). Данный способ представления заметно упрощает работу с конечными разностями высших порядков.

Конечные разности применяются в [интерполяционном методе Ньютона](https://ru.wikipedia.org/wiki/%D0%98%D0%BD%D1%82%D0%B5%D1%80%D0%BF%D0%BE%D0%BB%D1%8F%D1%86%D0%B8%D0%BE%D0%BD%D0%BD%D1%8B%D0%B5_%D1%84%D0%BE%D1%80%D0%BC%D1%83%D0%BB%D1%8B_%D0%9D%D1%8C%D1%8E%D1%82%D0%BE%D0%BD%D0%B0). С конечными разностями связаны понятия [разделённых разностей](https://ru.wikipedia.org/wiki/%D0%A0%D0%B0%D0%B7%D0%B4%D0%B5%D0%BB%D0%B5%D0%BD%D0%BD%D0%B0%D1%8F_%D1%80%D0%B0%D0%B7%D0%BD%D0%BE%D1%81%D1%82%D1%8C) и [модуля непрерывности](https://ru.wikipedia.org/wiki/%D0%9C%D0%BE%D0%B4%D1%83%D0%BB%D1%8C_%D0%BD%D0%B5%D0%BF%D1%80%D0%B5%D1%80%D1%8B%D0%B2%D0%BD%D0%BE%D1%81%D1%82%D0%B8).

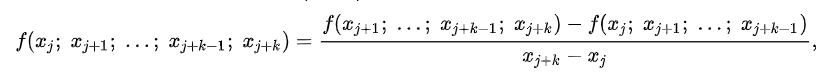
## Разделённые разности

**.**

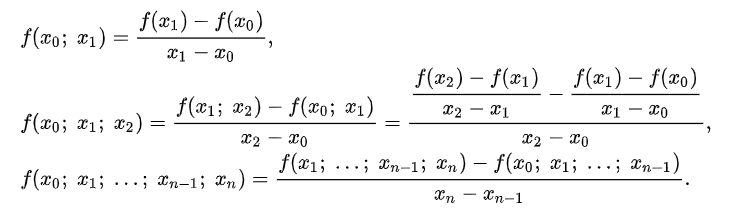
**Разделённая разность** — обобщение понятия [производной](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%BE%D0%B8%D0%B7%D0%B2%D0%BE%D0%B4%D0%BD%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%B8) для дискретного набора точек.

Пусть функция задана на (связном) множестве и фиксированы попарно различныеточки , ..., .

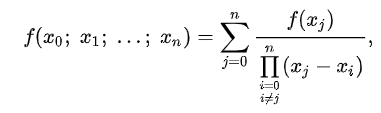
Тогда **разделённой разностью** **нулевого порядка** функции точке называют значение а **разделённую разность порядка** для системы точек определяют через разделённые разности порядка по формуле



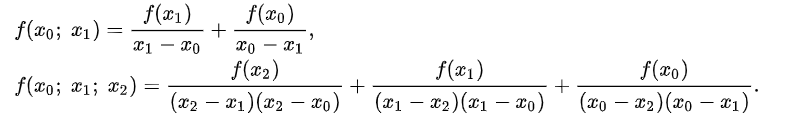
в частности,



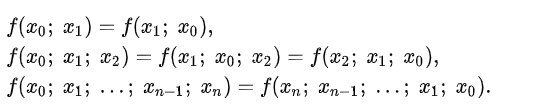
Для разделённой разности верна формула



в частности,



Разделённая разность является [симметрической функцией](https://ru.wikipedia.org/wiki/%D0%A1%D0%B8%D0%BC%D0%BC%D0%B5%D1%82%D1%80%D0%B8%D1%87%D0%B5%D1%81%D0%BA%D0%B8%D0%B9_%D0%BC%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD) своих аргументов, то есть при любой их перестановке её значение не меняется, в частности,



При фиксированной системе точек разность является [линейным функционалом](https://ru.wikipedia.org/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D1%8B%D0%B9_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D0%BE%D0%BD%D0%B0%D0%BB), то есть для функций и скаляров и :



## Интерполяционный метод Лагранжа

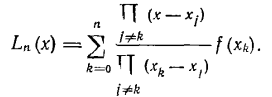
**.**

Интерполяционный многочлен Лагранжа — многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для *n + 1* пар чисел , где все различны, существует единственный многочлен *L(x)* степени не более *n*, для которого *L() =* .

В простейшем случае (*n = 1*) — это линейный многочлен, график которого — прямая, проходящая через две заданные точки.

Задача интерполирования многочленами состоит в том, чтобы построить многочлен:

значения которого в заданных точках *x* совпадают со значениями функции *f(x)* в этих точках.

Формула для нахождения многочлена:

где и — заданные точки (узлы).

## Интерполяционная формула Ньютона

**.**

Данная формула позволяет выразить интерполяционный многочлен через значение в одном из узлов и через разделенные разности функции , построенные по узлам . Она является разностным аналогом формулы Тейлора.

Формула разделённых разностей:

Интерполяционным многочленом Ньютона называется многочлен:

## Интерполяция сплайнами

**.**

**Сплайн** — [функция](https://ru.wikipedia.org/wiki/%D0%A4%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F_(%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0)), область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым алгебраическим [многочленом](https://ru.wikipedia.org/wiki/%D0%9C%D0%BD%D0%BE%D0%B3%D0%BE%D1%87%D0%BB%D0%B5%D0%BD) ([полиномом](https://ru.wikipedia.org/wiki/%D0%9F%D0%BE%D0%BB%D0%B8%D0%BD%D0%BE%D0%BC)). Максимальная из степеней использованных полиномов называется **степенью сплайна**. Разность между степенью сплайна и получившейся [гладкостью](https://ru.wikipedia.org/wiki/%D0%93%D0%BB%D0%B0%D0%B4%D0%BA%D0%B0%D1%8F_%D1%84%D1%83%D0%BD%D0%BA%D1%86%D0%B8%D1%8F) называется **дефектом сплайна**. Например, непрерывная ломаная есть сплайн степени 1 и дефекта 1. В современном понимании сплайны — это решения многоточечных [краевых задач](https://ru.wikipedia.org/wiki/%D0%9A%D1%80%D0%B0%D0%B5%D0%B2%D0%B0%D1%8F_%D0%B7%D0%B0%D0%B4%D0%B0%D1%87%D0%B0) сеточными методами.

Другими словами сплайн — это кусочно заданная функция, то есть совокупность нескольких функций, каждая из которых задана на каком-то множестве значений аргумента, причём эти множества попарно непересекающиеся.

Сплайны имеют многочисленные применения как в математической теории, так и в [прикладной математике](https://ru.wikipedia.org/wiki/%D0%9F%D1%80%D0%B8%D0%BA%D0%BB%D0%B0%D0%B4%D0%BD%D0%B0%D1%8F_%D0%BC%D0%B0%D1%82%D0%B5%D0%BC%D0%B0%D1%82%D0%B8%D0%BA%D0%B0) (в частности, в разнообразных вычислительных программах). В частности, сплайны двух переменных интенсивно используются для задания поверхностей в различных системах [компьютерного моделирования](https://ru.wikipedia.org/wiki/%D0%9A%D0%BE%D0%BC%D0%BF%D1%8C%D1%8E%D1%82%D0%B5%D1%80%D0%BD%D0%BE%D0%B5_%D0%BC%D0%BE%D0%B4%D0%B5%D0%BB%D0%B8%D1%80%D0%BE%D0%B2%D0%B0%D0%BD%D0%B8%D0%B5). Сплайны двух аргументов называют би-сплайнами (например, бикубический сплайн), которые являются двумерными сплайнами, моделирующими поверхности. Их часто путают с [B-сплайнами](https://ru.wikipedia.org/wiki/B-%D1%81%D0%BF%D0%BB%D0%B0%D0%B9%D0%BD) (базисными сплайнами), которые являются одномерными и в линейной комбинации составляют кривые — каркас для «натягивания» поверхностей. Также из базисных сплайнов возможно составить трёхмерную конструкцию для моделирования объёмных тел.

# Листинги программ и результаты

## Задание 1

**.**

Для заданных на отрезке функций

1. , , , , ;
2. , , , , ;
3. , , , , , ;
4. , , , , , ;

используйте три различных метода аппроксимации для оценки , , , , , . Сравните результаты по абсолютной ошибке.

## Код программы и результаты к заданию 1

**:**

import numpy as np  
import matplotlib.pyplot as plt  
from math import log, sin, cos, exp  
import math  
  
  
def right\_derivative(f, x, n, delta):  
 if n == 1:  
 return (f(x + delta) - f(x)) / delta  
 return (right\_derivative(f, x + delta, n - 1, delta) - right\_derivative(f, x, n - 1, delta)) / delta  
  
  
def left\_derivative(f, x, n, delta):  
 if n == 1:  
 return (f(x) - f(x - delta)) / delta  
 return (left\_derivative(f, x, n - 1, delta) - left\_derivative(f, x - delta, n - 1, delta)) / delta  
  
  
def derivative(f, x, n, delta):  
 if n == 1:  
 return (f(x + delta) - f(x - delta)) / (2 \* delta)  
 return (derivative(f, x + delta, n - 1, delta) - derivative(f, x - delta, n - 1, delta)) / (2 \* delta)  
  
  
z1 = [0.25, 0.51, 0.99, 1.09, 1.89, 2.39]  
delta = 0.001  
  
  
#  
def f1(x):  
 return exp(2 \* x) \* cos(3 \* x)  
  
  
print('e^(2x)\*cos(3x)')  
print("f(x)': 0.25) -0.9587949645")  
print("f(x)': 0.51) -8.0864523832")  
print("f(x)'': 0.99) 20.8413434406")  
print("f(x)'': 1.09) 57.4611173972")  
print("f(x)''': 1.89) -1421.4099360944")  
print("f(x)''': 2.39) 4292.7866586904")

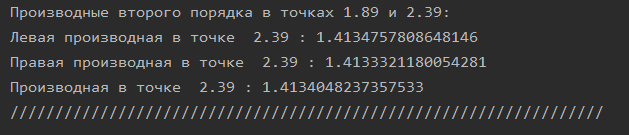
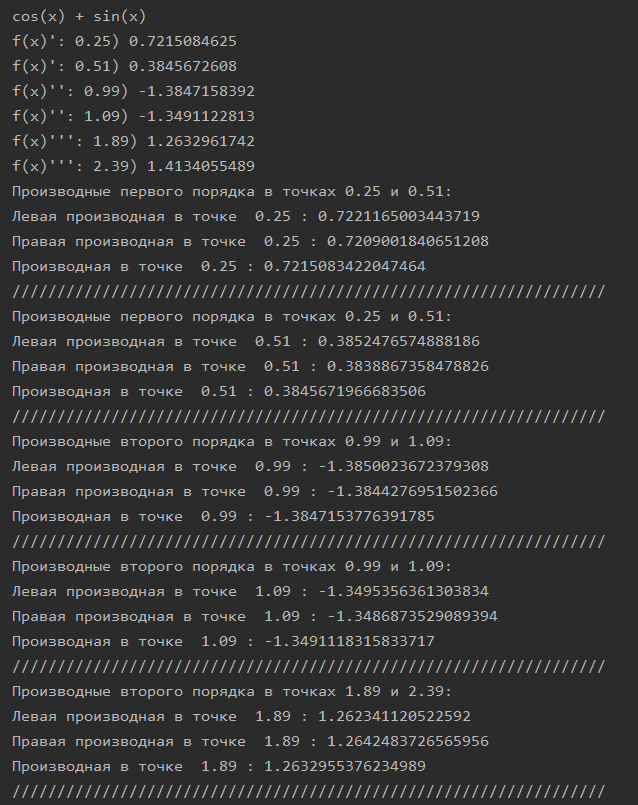
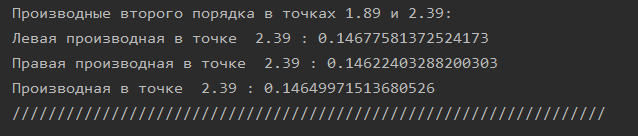
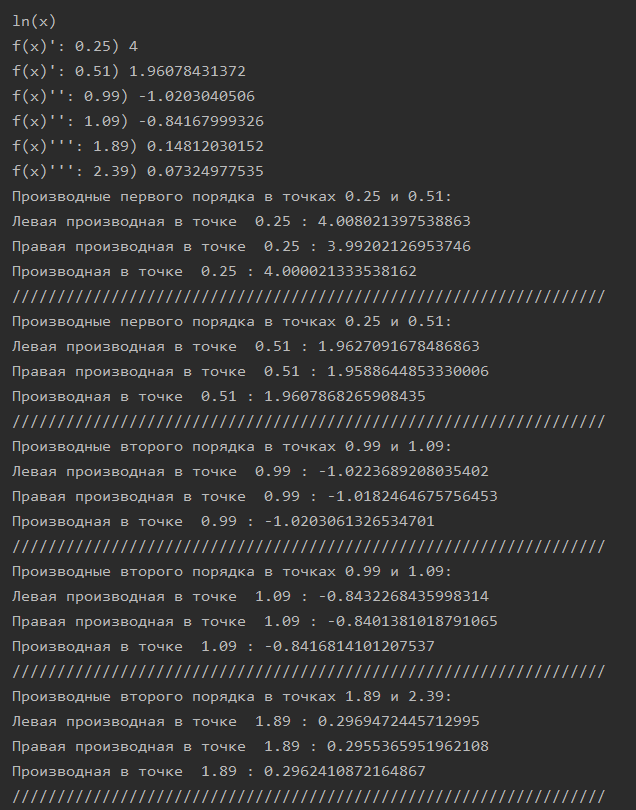
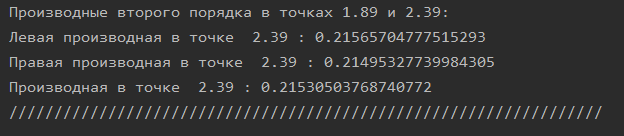
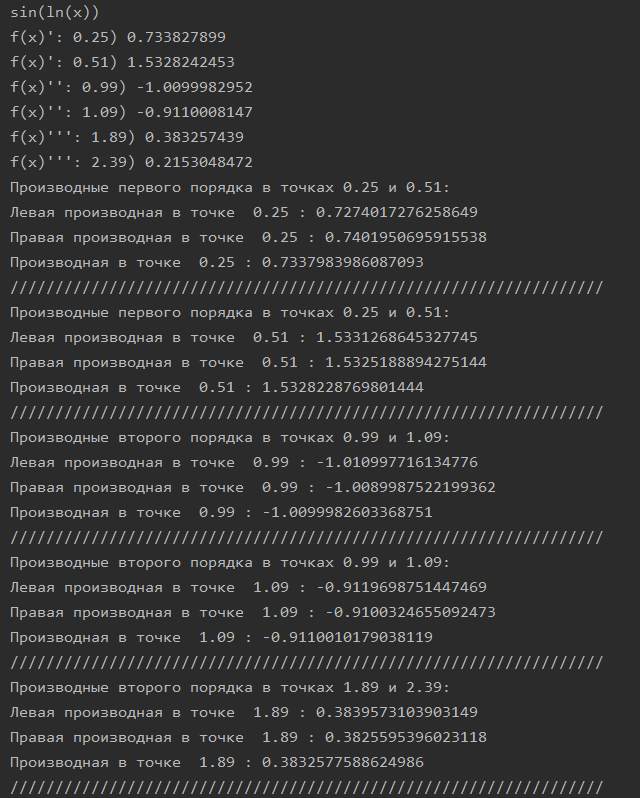
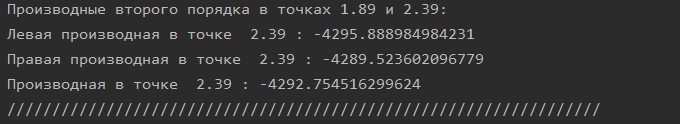
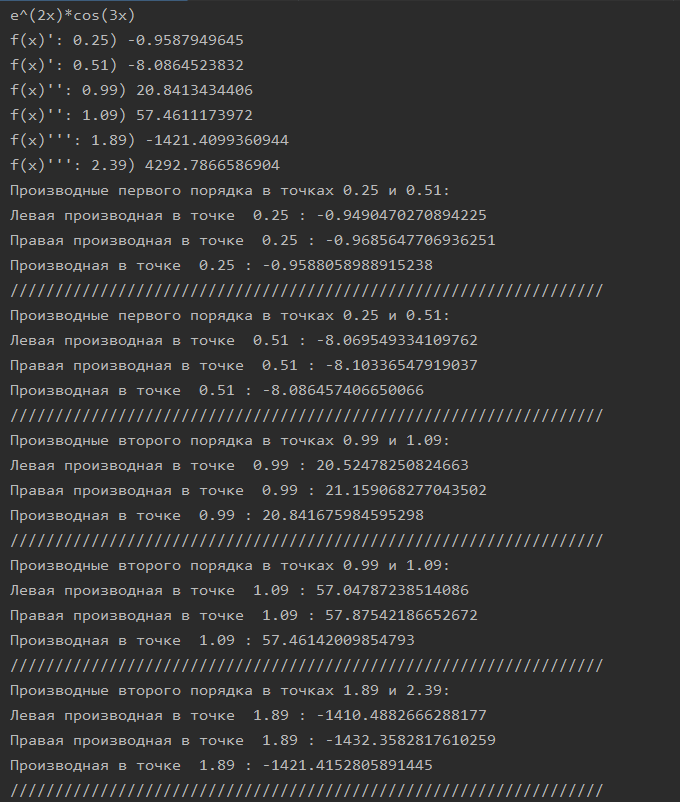
for dot in z1[:2]:  
 print('Производные первого порядка в точках 0.25 и 0.51:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f1, dot, 1, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f1, dot, 1, delta))  
 print('Производная в точке ', dot, ':', derivative(f1, dot, 1, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z1[2:4]:  
 print('Производные второго порядка в точках 0.99 и 1.09:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f1, dot, 2, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f1, dot, 2, delta))  
 print('Производная в точке ', dot, ':', derivative(f1, dot, 2, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z1[4:6]:  
 print('Производные второго порядка в точках 1.89 и 2.39:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f1, dot, 3, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f1, dot, 3, delta))  
 print('Производная в точке ', dot, ':', derivative(f1, dot, 3, delta))  
 print("//////////////////////////////////////////////////////////////////")  
print()  
  
z2 = [0.25, 0.51, 0.99, 1.09, 1.89, 2.39]  
  
  
def f2(x):  
 return sin(log(x))  
  
  
print('sin(ln(x))')  
print("f(x)': 0.25) 0.733827899")  
print("f(x)': 0.51) 1.5328242453")  
print("f(x)'': 0.99) -1.0099982952")  
print("f(x)'': 1.09) -0.9110008147")  
print("f(x)''': 1.89) 0.383257439")  
print("f(x)''': 2.39) 0.2153048472")  
for dot in z2[:2]:  
 print('Производные первого порядка в точках 0.25 и 0.51:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f2, dot, 1, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f2, dot, 1, delta))  
 print('Производная в точке ', dot, ':', derivative(f2, dot, 1, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z2[2:4]:  
 print('Производные второго порядка в точках 0.99 и 1.09:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f2, dot, 2, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f2, dot, 2, delta))  
 print('Производная в точке ', dot, ':', derivative(f2, dot, 2, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z2[4:6]:  
 print('Производные второго порядка в точках 1.89 и 2.39:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f2, dot, 3, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f2, dot, 3, delta))  
 print('Производная в точке ', dot, ':', derivative(f2, dot, 3, delta))  
 print("//////////////////////////////////////////////////////////////////")  
print()  
  
z3 = [0.25, 0.51, 0.99, 1.09, 1.89, 2.39]

def f3(x):  
 return log(x)  
  
  
print('ln(x)')  
print("f(x)': 0.25) 4")  
print("f(x)': 0.51) 1.96078431372")  
print("f(x)'': 0.99) -1.0203040506")  
print("f(x)'': 1.09) -0.84167999326")  
print("f(x)''': 1.89) 0.14812030152")  
print("f(x)''': 2.39) 0.07324977535")  
for dot in z3[:2]:  
 print('Производные первого порядка в точках 0.25 и 0.51:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f3, dot, 1, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f3, dot, 1, delta))  
 print('Производная в точке ', dot, ':', derivative(f3, dot, 1, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z3[2:4]:  
 print('Производные второго порядка в точках 0.99 и 1.09:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f3, dot, 2, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f3, dot, 2, delta))  
 print('Производная в точке ', dot, ':', derivative(f3, dot, 2, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z3[4:6]:  
 print('Производные второго порядка в точках 1.89 и 2.39:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f3, dot, 3, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f3, dot, 3, delta))  
 print('Производная в точке ', dot, ':', derivative(f3, dot, 3, delta))  
 print("//////////////////////////////////////////////////////////////////")  
print()  
  
z4 = [0.25, 0.51, 0.99, 1.09, 1.89, 2.39]  
  
  
def f4(x):  
 return cos(x) + sin(x)  
  
  
print('cos(x) + sin(x)')  
print("f(x)': 0.25) 0.7215084625")  
print("f(x)': 0.51) 0.3845672608")  
print("f(x)'': 0.99) -1.3847158392")  
print("f(x)'': 1.09) -1.3491122813")  
print("f(x)''': 1.89) 1.2632961742")  
print("f(x)''': 2.39) 1.4134055489")  
for dot in z4[:2]:  
 print('Производные первого порядка в точках 0.25 и 0.51:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f4, dot, 1, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f4, dot, 1, delta))  
 print('Производная в точке ', dot, ':', derivative(f4, dot, 1, delta))  
 print("//////////////////////////////////////////////////////////////////")  
for dot in z4[2:4]:  
 print('Производные второго порядка в точках 0.99 и 1.09:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f4, dot, 2, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f4, dot, 2, delta))  
 print('Производная в точке ', dot, ':', derivative(f4, dot, 2, delta))

print("//////////////////////////////////////////////////////////////////")  
for dot in z4[4:6]:  
 print('Производные второго порядка в точках 1.89 и 2.39:')  
 print('Левая производная в точке ', dot, ':', left\_derivative(f4, dot, 3, delta))  
 print('Правая производная в точке ', dot, ':', right\_derivative(f4, dot, 3, delta))  
 print('Производная в точке ', dot, ':', derivative(f4, dot, 3, delta))  
 print("//////////////////////////////////////////////////////////////////")  
print()

## Результаты работы

**:**



## **Задание 2**

**.**

Исторические данные о дневном приросте количества заболевших в мире позволяют оценить динамику заболеваемости. Результатом вычисления являются полиномы – *S*(*x*)*,* коэффициенты которых получены методом наименьших квадратов и сплайнами (см. результаты лабораторных работ по теме 6). Вычислите общее количество смертей

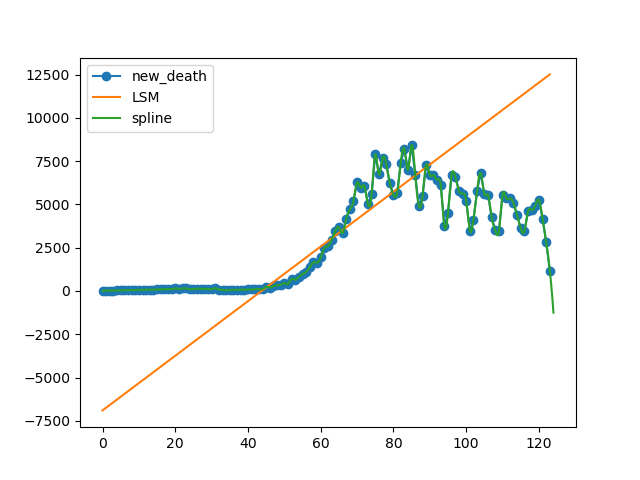
двумя различными численными методами. Сделайте выводы, сравнивая методы по точности, сходимости и метода оценки функции *S*(*x*)*.*

## Код программы и результаты к заданию 2:

import matplotlib.pyplot as plt  
import numpy as np  
import pandas as pd  
from math import sqrt  
from scipy.interpolate import CubicSpline  
  
  
def integral\_lsm(x\_i, y\_i):  
 result = 0  
 for i in range(0, len(x\_i) - 1):  
 result += (x\_i[i + 1] - x\_i[i]) \* (y\_i[i + 1] + y\_i[i]) / 2  
 return result  
  
  
def integral\_Gauss(x\_i, y\_i):  
 spline = CubicSpline(x\_i, y\_i)  
 result = 0  
 for i in range(0, len(x\_i) - 1):  
 temp1 = spline(((x\_i[i + 1] + x\_i[i]) + (x\_i[i + 1] - x\_i[i]) / sqrt(3)) / 2)  
 temp2 = spline(((x\_i[i + 1] + x\_i[i]) - (x\_i[i + 1] - x\_i[i]) / sqrt(3)) / 2)  
 result += (x\_i[i + 1] - x\_i[i]) \* (temp2 + temp1) / 2  
 return result  
  
  
def Draw(x\_i, y\_i):  
 n = len(x\_i)  
 sumx, sumy = sum(x\_i), sum(y\_i)  
 sX2 = sum([i \*\* 2 for i in x\_i])  
 a = n \* sum([(x\_i[i] \* y\_i[i]) for i in range(0, len(x\_i))])  
 a -= (sumx \* sumy)  
 a /= (n \* sX2 - sumx \*\* 2)  
 b = (sumy - a \* sumx) / n  
 mnk = np.array([b + a \* x\_i[i] for i in range(0, len(x\_i))])  
 plt.plot(x\_i, y\_i, marker='o', label="new\_death")  
 plt.plot(x\_i, mnk, label='LSM')  
  
  
def visualisation(x\_i, y\_i):  
 Draw(x\_i, y\_i)  
 spline = CubicSpline(x\_i, y\_i)  
 x = np.linspace(0, 124, 1000)  
 plt.plot(x, spline(x), label='spline')  
  
  
if \_\_name\_\_ == "\_\_main":  
 data = pd.read\_excel('E:/Countries-Confirmed.xlsx', header=0)

x = np.arange(0, 124, 1)  
 y = data['0204'][::-1]  
 visualisation(x, y)  
 lsm = integral\_lsm(x, y)  
 Gauss = integral\_Gauss(x, y)  
 print('метод наименьших квадратов:', lsm)  
 print('Точность метода:', 100 - lsm / 347596, '%')  
 print('метод Гаусса (с помощью сплайнов):', Gauss)  
 print('Точность метода:', 100 - Gauss / 347596, '%')  
 plt.legend()  
 plt.show()

## Результаты работы:



# Вывод

В процессе выполнения данной лабораторной работы нами были изучены методы интерполяции и аппроксимации.