Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и кибербезопасности

Высшая школа компьютерных технологий и информационных систем

**Отчет по лабораторной работе №1**

**Дисциплина: «**Практикум по вычислительной математике».

Выполнил

студент гр. 5130901/20003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ М. Д. Редько

(подпись)

Преподаватель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Цыган

(подпись)

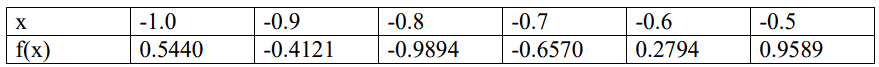
«\_\_\_» \_\_\_\_\_\_\_\_\_\_\_\_\_\_ 2024 г.

Санкт-Петербург   
2024

**ЗАДАНИЕ**

**Вариант 36:**

Для таблично заданной функции f(x):



построить сплайн-функцию и использовать ее для нахождения корня уравнения f(x) = 1.8x2 на промежутке [-1 , -0.5] методом бисекции. С использованием программы **QUANC8** вычислить интегралы от двух интерполирующих функций на интервале [-1 , -0.5] и сравнить их значения.

**ХОД РАБОТЫ**

Для выполнения работы был использован язык программирования Python версии 3.11 и следующие библиотеки:

* SciPy для вычислений.
* NumPy для работы с массивами точек.
* Matplotlib для представления данных в виде графиков.
* PrettyTable для представления данных в виде таблицы.

Во время работы были поставлены следующие цели:

1. По заданной таблице построить сплайн-функцию и полином Лагранжа
2. С использованием полученной на предыдущем шаге сплайн функции и метода бисекции, найти корень уравнения: f(x) = 1.8x2.
3. С помощью программы QUANC8 вычислить значения интегралов от двух интерполирующих функций, и сравнить их между собой.
4. Вычислить отклонения значений в узлах таблицы двух интерполирующих функций от табличных значений.

Сплайн

Сплайн функцию можно построить с помощью функции CubicSpline, из библиотеки SciPy. Эта функция возвращает коэффициенты b, c и d:

# Вычисление коэффициентов сплайн-функции  
spline\_coeff = CubicSpline(x\_table\_values, f\_table\_values)

Полином Лагранжа

Полином в форме Лагранжа можно получить с помощью функцииlagrange, из все той же библиотеки SciPy:

# Получаем полином Лагранжа  
lagrange\_coeff = lagrange(x\_table\_values, f\_table\_values)

Бисекция

В библиотеке SciPy имеется встроенноя функция, реализующая метод бисекции – bisect:

# Определение функции 1.8 \* x^2  
def f(x):  
 return 1.8 \* (x \*\* 2)

# Функция: spline(x) - 1.8 \* x^2 = 0  
def f\_spline(x):  
 return spline\_values(x) - f(x)

# Находим корень уравнения f(x) = 1.8 \* x^2 методом бисекции  
root = bisect(f\_spline, a=a, b=b)

В данном методе будем использовать погрешность по умолчанию - 4.4408920985006262e-16.

QUANC8

Функции QUANC8 нет в библиотеке SciPy, но есть похожая – quad, задав ограничение в 30 шагов для последней, получаем аналог функции QUANC8:

# Функция, реализирующая функцию QUANC8  
def QUANC8(func):  
 return quad(func=func, a=a, b=b, limit=30)

Полный код программы прилагается.

**РЕЗУЛЬТАТЫ РАСЧЕТОВ**

Для «удобного» и «красивого» вывода использовались библиотеки Matplotlib и PrettyTable, функционал которых дает возможность выводить графики и таблицы:

Графики табличной функции, 1.8\*x2, и двух интерполирующих функций:

Изображение выглядит как текст, График, снимок экрана, линия

Автоматически созданное описание

Как видно, сплайн и полином Лагранжа почти сливаются, но различимы расхождения в начале интервала [-1.0; -0.5], обе эти функции схожи с характером функции из начальной таблицы.

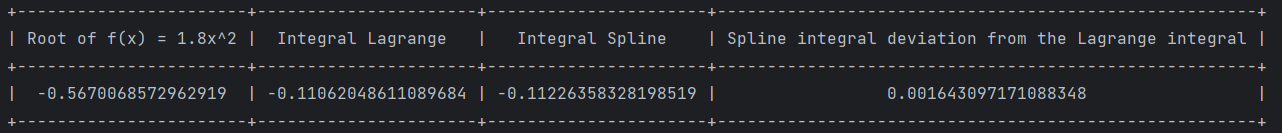
Далее численные результаты:

Изображение выглядит как снимок экрана, текст, Шрифт

Автоматически созданное описаниеВ представленной таблице можно наблюдать значения в узлах таблицы сплайна и полинома Лагранжа, их отклонение от табличных, а также, отклонение друг от друга (сплайна от полинома Лагранжа)

Как видно, обе интерполирующие функции имеют относительно точное приближение к исходным данным, в случае сплайна – полное совпадение почти во всех узлах.

Далее еще одна таблица:



В ней можно видеть вычисленные интегралы от двух интерполирующих функций и разность между этими интегралами, а так же – корень уравнения f(x) = 1.8x2

Напоследок еще один график – погрешности полинома Лагранжа и слайн-функции:

Изображение выглядит как текст, снимок экрана, График, линия

Автоматически созданное описание

Как упоминалось ранее, сплайн функция в данном случае выглядит более точной.

**ВЫВОД**

В ходе выполнения данной работы, был изучен функционал математических библиотек, для языка программирования Python, и применен на практике. И полученных результатов можно сделать вывод – в нашем случае сплайн функция интерполирует лучше полинома Лагранжа на всем заданном промежутке. И хоть их отклонение друг от друга почти неразличимо визуально, за счет малых порядков погрешности (~10-13-10-16), нельзя исключать случаи, где требуется высокая точность, и даже такая «малая» ошибка может повлечь за собой «большие» последствия.

**ПРИЛОЖЕНИЕ**

import numpy as np  
from prettytable import PrettyTable  
from scipy.interpolate import CubicSpline, lagrange  
from scipy.integrate import quad  
from scipy.optimize import bisect  
import matplotlib.pyplot as plt  
  
# Исходные табличные данные  
x\_table\_values = np.array([-1.0, -0.9, -0.8, -0.7, -0.6, -0.5])  
f\_table\_values = np.array([0.5440, -0.4121, -0.9894, -0.6570, 0.2794, 0.9589])  
a = -1.0  
b = -0.5  
  
# Вычисление коэффициентов сплайн-функции  
spline\_coeff = CubicSpline(x\_table\_values, f\_table\_values)  
  
# Получаем полином Лагранжа  
lagrange\_coeff = lagrange(x\_table\_values, f\_table\_values)  
  
  
# Функция, вычисляющая значение полинома Лагранжа в точке x  
def lagrange\_values(x):  
 return lagrange\_coeff(x)  
  
  
# Функция, вычисляющая значение сплайн-функции в точке x  
def spline\_values(x):  
 return spline\_coeff(x)  
  
  
# Определение функции 1.8 \* x^2  
def f(x):  
 return 1.8 \* (x \*\* 2)  
  
  
# Функция: spline(x) - 1.8 \* x^2 = 0  
def f\_spline(x):  
 return spline\_values(x) - f(x)  
  
  
# Находим корень уравнения f(x) = 1.8 \* x^2 методом бисекции  
root = bisect(f\_spline, a=a, b=b)  
  
  
# Функция, реализирующая функцию QUANC8  
def QUANC8(func):  
 return quad(func=func, a=a, b=b, limit=30)  
  
  
# Вычисление интегралов с использованием QUANC8  
integral\_spline, error\_1 = QUANC8(spline\_coeff)  
integral\_lagrange, error\_2 = QUANC8(lagrange\_coeff)  
  
# Создание таблицы для вывода результатов вычислений  
  
lagrange\_deviation = f\_table\_values - lagrange\_values(x\_table\_values)  
spline\_deviation = f\_table\_values - spline\_values(x\_table\_values)  
  
table\_one = PrettyTable()  
table\_one.add\_column("x", x\_table\_values)  
table\_one.add\_column("f(x)", f\_table\_values)  
table\_one.add\_column("Lagrange", lagrange\_values(x\_table\_values))  
table\_one.add\_column("Spline", spline\_values(x\_table\_values))  
table\_one.add\_column("Lagrange deviation from f(x)", lagrange\_deviation)  
table\_one.add\_column("Spline deviation from f(x)", spline\_deviation)  
table\_one.add\_column("Spline deviation from the Lagrange",  
 abs(lagrange\_values(x\_table\_values) - spline\_values(x\_table\_values)))  
  
table\_two = PrettyTable()  
table\_two.add\_column("Root of f(x) = 1.8x^2", [root])  
table\_two.add\_column("Integral Lagrange", [integral\_lagrange])  
table\_two.add\_column("Integral Spline", [integral\_spline])  
table\_two.add\_column("Spline integral deviation from the Lagrange integral", [abs(integral\_spline - integral\_lagrange)])  
  
print(table\_one)  
print(table\_two)  
  
x\_values = np.linspace(-1.0, -0.5, 100)  
  
# Вывод функций графически  
plt1 = plt  
plt1.plot(x\_table\_values, f\_table\_values, label='Table function')  
plt1.plot(x\_values, spline\_values(x\_values), label='Spline Function')  
plt1.plot(x\_values, f(x\_values), label='1.8x^2 function')  
plt1.plot(x\_values, lagrange\_values(x\_values), label='Lagrange function')  
plt1.xlabel('x')  
plt1.ylabel('f(x)')  
plt1.legend()  
plt1.show()  
  
# Отображение графиков погрешностей  
plt2 = plt  
plt2.plot(x\_table\_values, abs(lagrange\_deviation), label='Lagrange deviation')  
plt2.plot(x\_table\_values, spline\_deviation, label='Spline deviation')  
plt2.xlabel('x')  
plt2.legend()  
plt2.show()