Санкт-Петербургский политехнический университет Петра Великого

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Отчёт по лабораторной работе № 1**

Дисциплина: Вычислительная математика

Выполнил студент гр. 3530901/10001 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Д.Л. Симоновский

(подпись)

Руководитель \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ В.Н. Цыган

(подпись)

“09” февраль 2023 г.

Санкт-Петербург

2023

Оглавление

[Задание: 2](#_Toc126883290)

[Инструменты: 2](#_Toc126883291)

[Ход выполнения работы: 2](#_Toc126883292)

[***Порядок действий:*** 2](#_Toc126883293)

[***Первая задача:*** 2](#_Toc126883294)

[Полная формулировка: 2](#_Toc126883295)

[Разбиваем на задачи: 2](#_Toc126883296)

[Пункт 1: 2](#_Toc126883297)

[Пункт 2: 3](#_Toc126883298)

[Пункт 3: 3](#_Toc126883299)

[***Вторая задача:*** 5](#_Toc126883300)

[Полная формулировка: 5](#_Toc126883301)

[Проблемы: 5](#_Toc126883302)

[Решение: 5](#_Toc126883303)

# Задание:

Вариант 11:

Для функции по узлам построить полином Лагранжа L(x) 10-й степени и сплайн-функцию S(x). Вычислить значения всех трех функций в точках . Результаты отобразить графически.   
Используя программу QUANC8, вычислить два интеграла: , для   
m = -1 и для m = -0.5.

# Инструменты:

Для работы был выбран язык программирования Python версии 3.11 по причине удобства его использования для поставленной задачи. Были выбраны следующие библиотеки:

* NumPy – для большей скорости расчетов
* SciPy – для функций расчета интерполяции и интеграла
* PrettyTable – для красивого вывода в консоль таблицы
* MatplotLib – для вывода графиков

# Ход выполнения работы:

## ***Порядок действий:***

Поставленное задание легко можно разбить на две глобальные задачи:

1. Выполнить интерполяцию
2. Выполнить расчет интеграла при m = -1, -0.5

## ***Первая задача:***

### Полная формулировка:

Для функции по узлам построить полином Лагранжа L(x) 10-й степени и сплайн-функцию S(x). Вычислить значения всех трех функций в точках . Результаты отобразить графически.

### Разбиваем на задачи:

В поставленной задаче явно есть несколько этапов:

1. Построить полином Лагранжа и сплайн-функцию
2. Вычислить значения в точках
3. Вывести результаты графически

### Пункт 1:

Создадим узлы интерполяции. Для этого воспользуемся функцией NumPy, который при добавлении мы переименовали в np для удобства:

x\_interpolation = np.arange(0, 1 + 0.1, 0.1)

y\_interpolation = 1 / (1 + x\_interpolation)

Далее выполним интерполяцию, используя функции библиотеки SciPy:

lagrange\_interpolation = lagrange(x\_interpolation, y\_interpolation

spline\_interpolation = CubicSpline(x\_interpolation, y\_interpolation)

### Пункт 2:

Аналогично узлам интерполяции создадим массив узлов, в которых будем считать погрешность интерполирования и выводить графики:

x\_find = np.arange(0.05, 1, 0.1)

Далее необходимо найти значения полиномов в точках x\_find, для этого выполним:

y\_lagrange = lagrange\_interpolation(x\_find)

y\_spline = spline\_interpolation(x\_find)

### Пункт 3:

Для отрисовки рассчитаем реальное значение функции в этой точке:

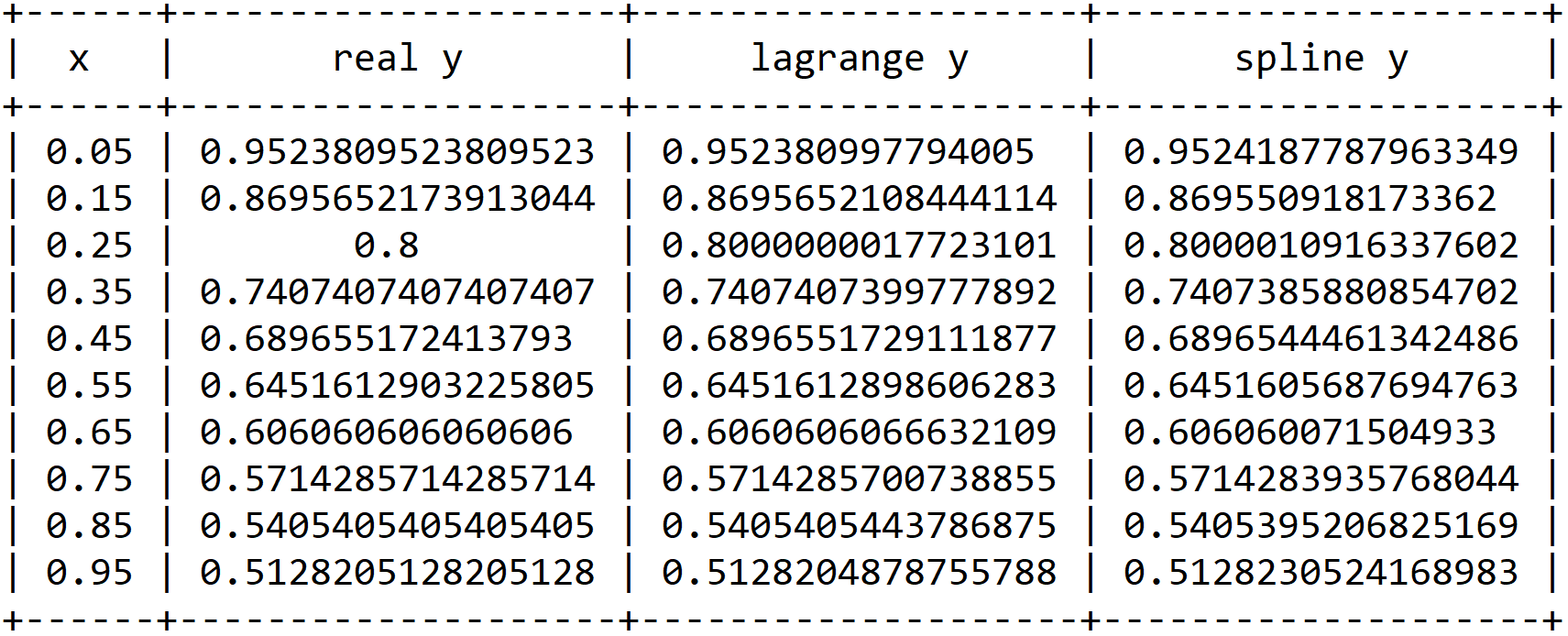
y\_real = 1 / (1 + x\_find)

Далее, в традициях функционального программирования, напишем функцию для отрисовки значений полиномов.

Напишем вывод полученных значений в консоль, используя библиотеку PrettyTable:

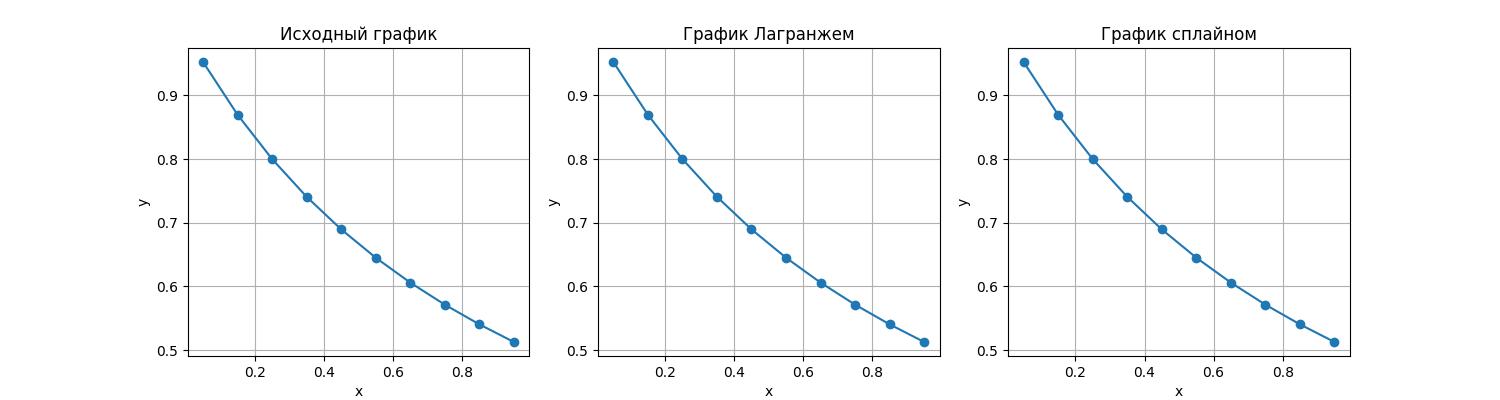
def print\_interpolation\_table(x\_find, y\_real, y\_lagrange, y\_spline):  
 pt = PrettyTable()  
 pt.add\_column('x', np.round(x\_find, 4))  
 pt.add\_column('real y', y\_real)  
 pt.add\_column('lagrange y', y\_lagrange)  
 pt.add\_column('spline y', y\_spline)  
 print(pt)

Полученный вывод:



Далее по заданию необходимо вывести графики. Сделаем это используя библиотеку MatplotLib, а конкретно модуль pyplot, переименуем его в plt.

def print\_one\_graph(x, y, title, id, count\_graphs):  
 plt.subplot(1, count\_graphs, id)  
 plt.xlabel('x')  
 plt.ylabel('y')  
 plt.grid()  
 plt.title(title)  
 plt.plot(x, y, '-o')  
  
  
def print\_interpolation\_graph(x\_find, y\_real, y\_lagrange, y\_spline):  
 plt.figure(figsize=(15, 4))  
 print\_one\_graph(x\_find, y\_real, 'Исходный график', 1, 3)  
 print\_one\_graph(x\_find, y\_lagrange, 'График Лагранжем', 2, 3)  
 print\_one\_graph(x\_find, y\_spline, 'График сплайном', 3, 3)  
 plt.savefig("Graphs.jpg")  
 plt.show()

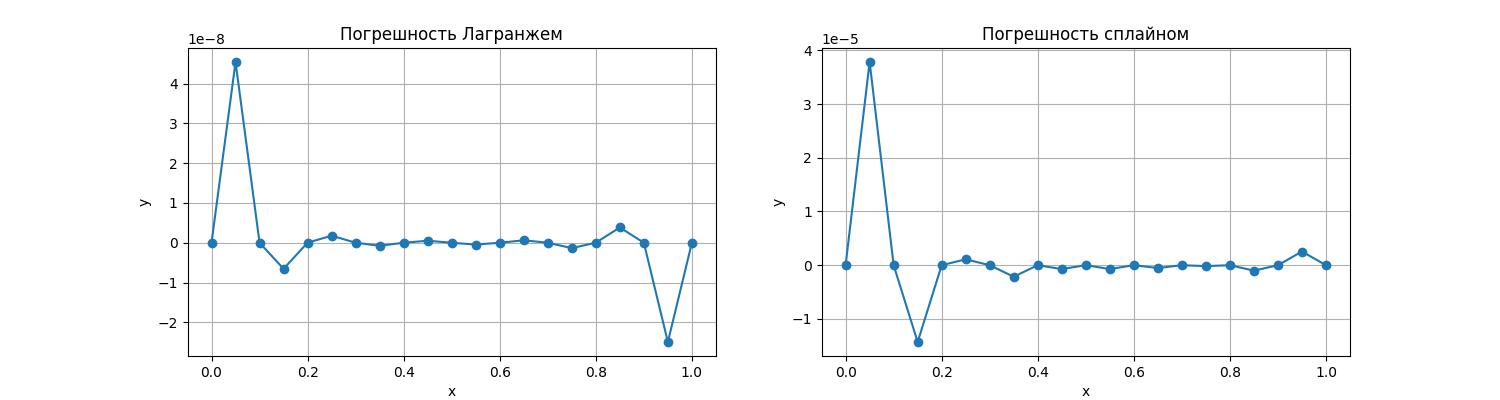
Полученные графики:  


Графики. Graphs.jpg

Выведем график погрешности т.к. на самих графиках не видно разницы. Будем использовать те же модули:

def print\_interpolation\_error(x\_interpolation, x\_find, y\_lagrange\_error, y\_spline\_error):  
 plt.figure(figsize=(15, 4))  
 # Добавляем узлы т.к. при интерполяции в них погрешность нулевая  
 x = np.arange(0, 1 + 0.05, 0.05)  
 # Добавляем нулевые узлы на график  
 y\_lagrange = [0 if i % 2 == 0 else y\_lagrange\_error[i // 2] for i in range(0, len(x\_interpolation) + len(x\_find))]  
 y\_spline = [0 if i % 2 == 0 else y\_spline\_error[i // 2] for i in range(0, len(x\_interpolation) + len(x\_find))]  
 # Собственно сам график  
 print\_one\_graph(x, y\_lagrange, 'Погрешность Лагранжем', 1, 2)  
 print\_one\_graph(x, y\_spline, 'Погрешность сплайном', 2, 2)  
 plt.savefig("Error.jpg")  
 plt.show()

Полученные графики:

  
Графики погрешности. Error.jpg

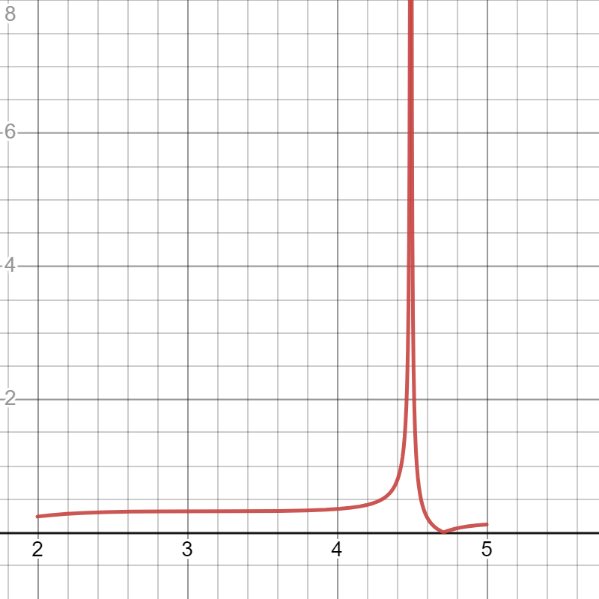
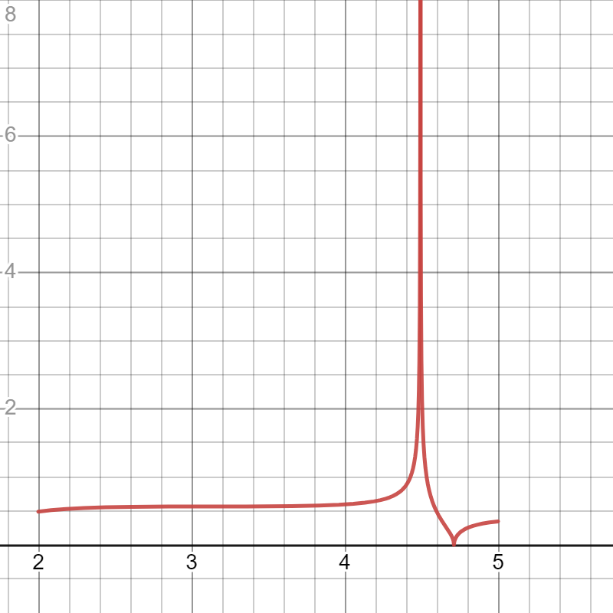
## ***Вторая задача:***

### Полная формулировка:

Используя программу QUANC8, вычислить два интеграла: , для   
m = -1 и для m = -0.5.

### Проблемы:

Построим график подынтегральной функции в сторонних приложениях.

При m = -1: При m = -0.5  
 

Как видно из графиков у нас имеется точка, в которой функция уходит в бесконечность, её значение найдем путем решения уравнения , эта точка x = ~4.49340945790906.

При решении придется её отдельно обработать.

### Решение:

Создаем функцию, которая будет на основании m возвращать нам другую функцию для интегрирования:

def get\_integrate\_function(m):  
 def integrate\_function(x):  
 return (np.abs(x - np.tan(x))) \*\* m  
  
 return integrate\_function

Так же т.к. в Python нет реализации QUANC8, возьмем аналог, основанный на адаптивных квадратурных формах, опирающийся на методы из библиотеки QUADPACK, языка Fortran:

def pseudo\_quanc8(func, start, end):  
 return quad(func, start, end, limit=30)

Используя эти функции напишем функцию для интегрирования нашего выражения:

def do\_integrate(m, approximation):  
 inconvenient\_point = 4.49340945790906  
 function = get\_integrate\_function(m)  
 integral = pseudo\_quanc8(function, 2, inconvenient\_point - approximation) + \  
 pseudo\_quanc8(function, inconvenient\_point + approximation, 5)  
 return integral[0]

Для получения значения интегралов вызовем этот метод 2 раза и выведем значения на экран:

def integration():  
 print(f'Integral with m=-1: {do\_integrate(-1.0, 2e-9)}')  
 print(f'Integral with m=-0.5: {do\_integrate(-0.5, 1e-6)}')

Получим   
Integral with m=-1: 1.6997777239732015

Integral with m=-0.5: 1.5276545405394897

Однако эти значения являются примерными т.к. мы не имеем возможности посчитать бесконечное значение у графика, точность, насколько мы приближаемся к пику мы высчитываем путем перебора значений приближения.

# Вывод:

В ходе работы я ознакомился с аналогами QUANC8, SPLINE, SEVAL на языке Python и получил опыт работы с ними, так же научились обрабатывать исключительные ситуации, как во втором задании. По результатам работы видно, что сплайн-функция приближает много лучше, чем полином Лагранжа, а также видно, что интерполяция тем лучше, чем дальше значение от краёв промежутка.

# Ссылки:

С листингом кода можно ознакомиться на [github.com](https://github.com/DafterT/comp_math_1)