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. 3530901/10003 \_\_\_\_\_\_\_\_\_\_\_\_\_\_ Я.А. Иванов

(подпись)

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

(подпись)

“27” марта 2023 г.

Санкт-Петербург

2023

Оглавление

[Задание: 2](#__RefHeading___Toc1279_301334972)

[Инструменты: 2](#__RefHeading___Toc1281_301334972)

[Ход выполнения работы: 2](#__RefHeading___Toc1283_301334972)

[*Порядок действий:* 2](#__RefHeading___Toc1285_301334972)

[*Первая задача:* 2](#__RefHeading___Toc1287_301334972)

[*Вторая задача:* 2](#__RefHeading___Toc1289_301334972)

[Третья задача 3](#__RefHeading___Toc1541_301334972)

[Вывод: 5](#__RefHeading___Toc1291_301334972)

[Ссылки: 5](#__RefHeading___Toc1293_301334972)

# Задание:

**Вариант 24:**

# Инструменты:

Для работы был выбран язык программирования Python версии 3.09 ввиду наличия необходимых библиотек для выполнения поставленной задачи, а именно:

* NumPy – для большей скорости расчетов
* SciPy – для функций расчета интерполяции и интеграла
* pandas – 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. Вычислим значение функции при помощи QUANC8 и по полученным точкам построим сплайн-функцию и полином Лагранжа 10-й степени
2. Вычислим значения в точках xk=1.1+0.2k в QUANC8 с высокой точностью и в сплайн-функции и полиноме и вычисление погрешности
3. Отображение полученных данных

## ***Первая задача:***

Зададим исходную функцию

def f(t):  
 return np.sqrt(t) \* np.sin(t)

Зададим x в интервале [1, 3] с шагом 0.2 при помощи библиотеки NumPy

x\_values = np.arange(1, 3.2, 0.2)

Вычислим значение функции при помощи QUANC8 и сохраним результат в integral\_values. В языке python нет готовой реализации QUANC8, однако в библиотеке SciPy есть ее аналог.

for i, x in enumerate(x\_values):  
 integral\_values[i], error = quad(f, 0, x, limit=30)

В переменной integral\_values находятся точки, вычисленные при помощи QUANC8. На основе этих точек построим функции:

spline\_func = interp1d(x\_values, integral\_values, kind='cubic')  
lagrange\_poly = P.Polynomial.fit(x\_values, integral\_values, 10)

Выполнение первой задачи завершено

## ***Вторая задача:***

Выполнив первый пункт у нас есть сплайн-функция и полином Лагранжа. Теперь сравним результат из работы со значениями QUANC8 с повышенной точностью. Для этого зададим x\_k = 1.1+0.2k где k=0,1,2,…,9

xk\_values = np.arange(1.1, 3.0, 0.2)

Повторно вызовем QUANC8, но на этот раз увеличим limit до 50.

quanc8\_values = np.zeros(len(xk\_values))  
for i, x in enumerate(xk\_values):  
 quanc8\_values[i], error = quad(f, 0, x, limit=50)

И вычислим значение в сплайн-функции и полиноме Лагранжа:

spline\_values = spline\_func(xk\_values)  
lagrange\_values = lagrange\_poly(xk\_values)

Вычислим погрешность:

spline\_errors = np.abs(spline\_values - quanc8\_values)  
lagrange\_errors = np.abs(lagrange\_values - quanc8\_values)

Теперь можно переходить к выводу результатов работы программы.

## ***Третья задача***

Для начала выведем результаты в виде таблицы. Для этого воспользуемся библиотекой pandas.

results = pd.DataFrame({  
 'xk': xk\_values,  
 'QUANC8 value': quanc8\_values,  
 'Spline value': spline\_values,  
 'Lagrange value': lagrange\_values,  
 'Spline error': spline\_errors,  
 'Lagrange error': lagrange\_errors  
})  
  
print(results.to\_string(index=False))

Вывод:

xk QUANC8 value Spline value Lagrange value Spline error Lagrange error

1.1 0.453081 0.453013 0.453081 6.803166e-05 4.500999e-09

1.3 0.656992 0.657018 0.656992 2.663262e-05 6.298331e-10

1.5 0.889802 0.889801 0.889802 3.669581e-07 1.659812e-10

1.7 1.142206 1.142211 1.142206 4.979426e-06 7.000067e-11

1.9 1.402991 1.402993 1.402991 1.295905e-06 4.411138e-11

2.1 1.659626 1.659627 1.659626 3.915170e-07 4.037815e-11

2.3 1.898913 1.898910 1.898913 3.181653e-06 5.362844e-11

2.5 2.107675 2.107676 2.107675 1.027971e-06 1.059361e-10

2.7 2.273455 2.273433 2.273455 2.284447e-05 3.560743e-10

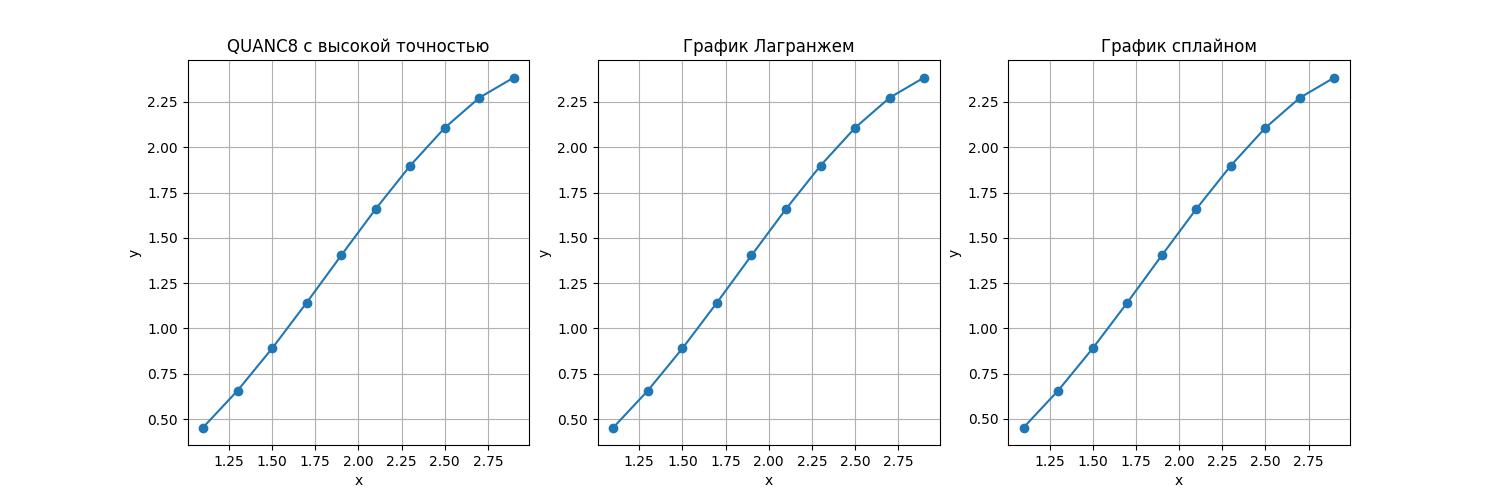
2.9 2.385184 2.385242 2.385184 5.869897e-05 2.012869e-09

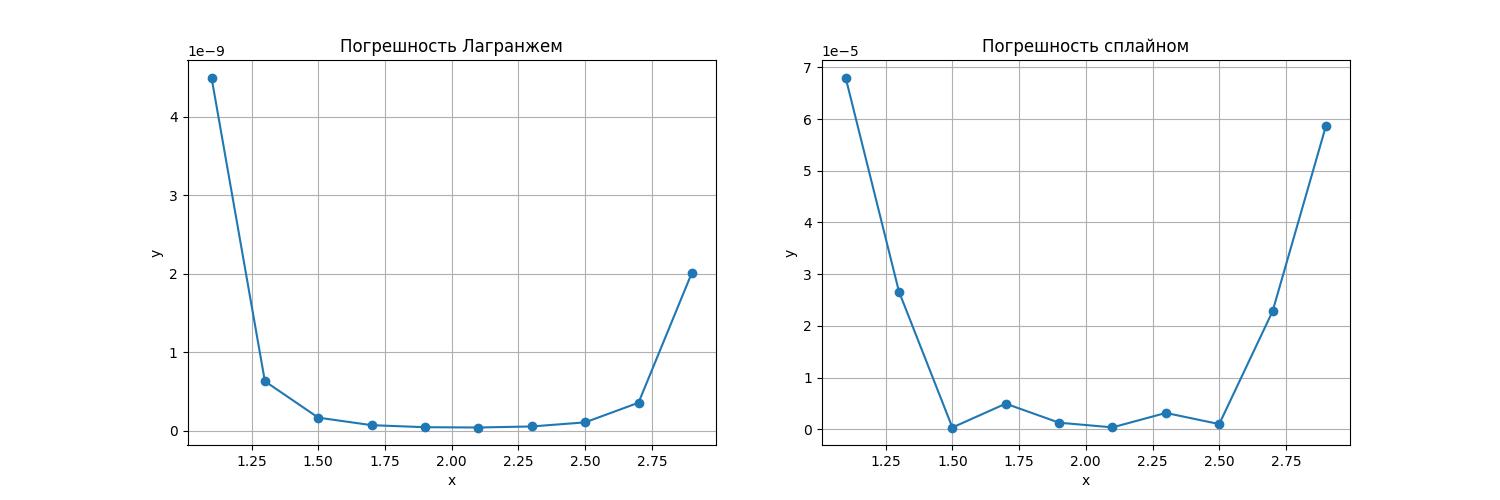
Далее представим графики значений функций, затем графики погрешности. Для этого воспользуемся библиотекой matplotlib:

def print\_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')

plt.figure(figsize=(15, 5))  
print\_graph(xk\_values, quanc8\_values, 'QUANC8 с высокой точностью', 1, 3)  
print\_graph(xk\_values, lagrange\_values, 'График Лагранжем', 2, 3)  
print\_graph(xk\_values, spline\_values, 'График сплайном', 3, 3)  
plt.savefig("Graphs.jpg")  
plt.show()  
  
plt.figure(figsize=(15, 5))  
print\_graph(xk\_values, lagrange\_errors, 'Погрешность Лагранжем', 1, 2)  
print\_graph(xk\_values, spline\_errors, 'Погрешность сплайном', 2, 2)  
plt.savefig("Error.jpg")  
plt.show()

На экран поочерёдно будут выведены следующие изображения:





# Вывод:

В ходе работы я ознакомился с аналогами QUANC8, SPLINE, полином Лагранжа на языке Python и получил опыт работы с ними, так же научились обрабатывать исключительные ситуации, как во втором задании. По результатам работы видно, что полином Лагража имеет меньшую погрешность, чем сплайн-функция при построении по точкам QUANC8.

# Ссылки:

Листин кода на github: <https://github.com/vitaya-para/lab1_2023/blob/main/main.py>