Федеральное государственное бюджетное образовательное учреждение высшего профессионального образования

«Санкт-Петербургский политехнический университет Петра Великого»

Институт компьютерных наук и технологий

Высшая школа интеллектуальных систем и суперкомпьютерных технологий

**Лабораторная работа №1**

по дисциплине «Вычислительная математика»

**Вариант №21**

Выполнил

Студент группы 3530901/80004 Иванов К. А.

\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Цыган В. Н.

\_\_\_\_\_\_\_\_\_\_\_

Санкт-Петербург 2020

**Постановка задачи:**

Для функции **f(x) = sin(x2)** по узлам **xk = 0.2k** (k = 0,1,..,6) **построить** полином **Лагранжа L(x)** 6-й степени и **сплайн-функцию S(x)**. **Вычислить** значения всех трех функций в точках **yk = 0.1 + 0.2k** (k = 0,1,…,5). Результаты отобразить графически.

Используя программу **QUANC8**, вычислить два интеграла:

, для m = -1 и для m = -0.5.

После прочтения задачи становится очевидным разбить задачу на две части и решить их отдельно, в таком виде и будем работать.

**Описание решения:**

1.Приступим к выполнению первой части задания, где нам нужно вычислить значения трех функций, сравнить их и отобразить графически

Для этого зададим необходимую функцию (с помощью метода в языке программирования, который при обращении возвращает заданную f(x)), зададим узлы xk с помощью листа(попросту массива) из 7 значений, тем самым, в дальнейшем будем иметь 6 промежутков, которые необходимы для построения полинома Лагранжа 6-й степени по ним. Вычисляем значения функции в заданных точках напрямую и через L(x) и сплайн-функцию S(x). Вычисления и построения функции и графиков будет выполнено при помощи двух популярных библиотек языка Python 3, SciPy и matplotlib.

Ознакомиться с тем, как работают внутренне та или иная операция можно на соответствующем библиотеке сайте с документацией.

SciPy:[docs.scipy.org/doc/scipy/reference/interpolate.html](https://docs.scipy.org/doc/scipy/reference/interpolate.html)

matplotlib:[matplotlib.org/api/\_as\_gen/matplotlib.pyplot.html?highlight=plot#module-matplotlib.pyplot](https://matplotlib.org/api/_as_gen/matplotlib.pyplot.html?highlight=plot#module-matplotlib.pyplot)

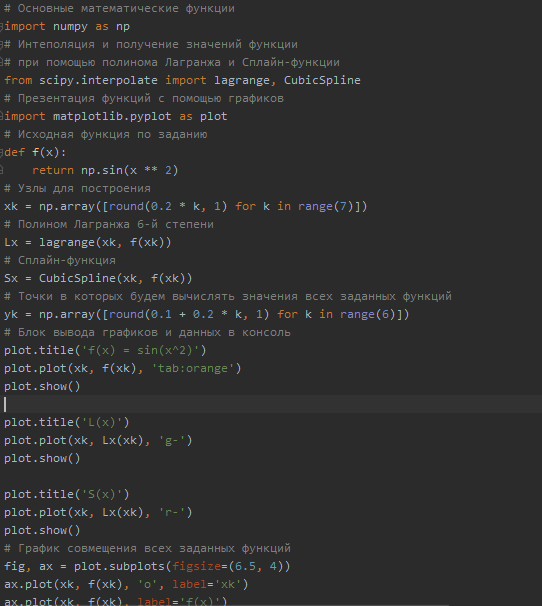
После подстановки точек yk и вычисления значений у всех трех функций, отобразим результаты графически, чтобы сравнить и увидеть разницу.

2. Во второй части задания нет ничего особенного, в начале оценим с какой подынтегральной функцией имеем дело, и обнаружим что на заданном промежутке [0, 2] функция имеет точку разрыва, вычислим интеграл с помощью программы QUANC8( в python – quad) с разной частотой приближений к точке разрыва, результаты отобразим в форме посчитанного интеграла и сравним их после получения результатов.

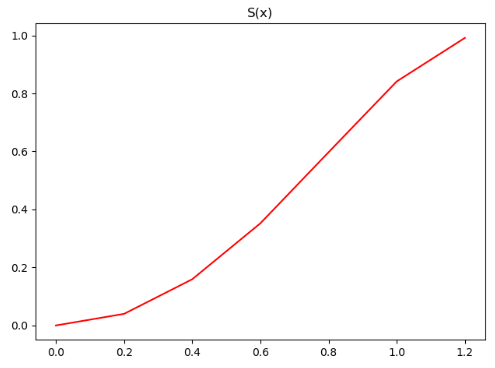
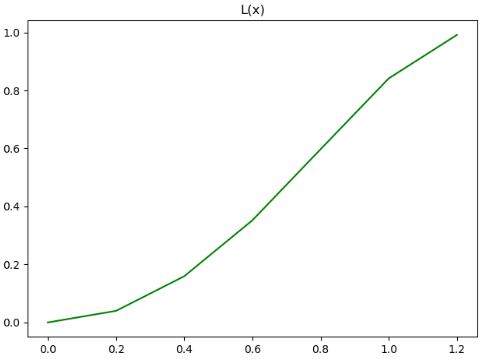
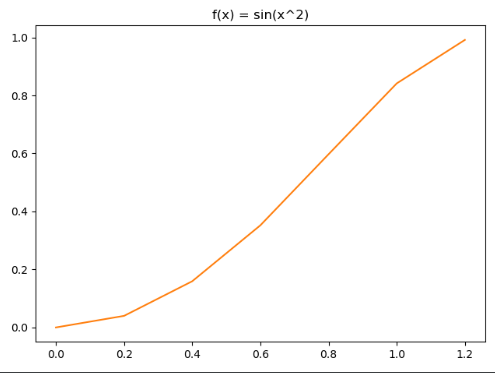
**Ход работы:**

**Original, Lagrange, Spline.**

Описываем исходную функцию f(x), добавляем нужные узлы в массив xk, строим полином Лагранжа Lx и сплайн-функцию, также добавляем массив точек yk, а дальше выводим графики, результаты функций в точках yk и сравним работу функций с исходной функций:

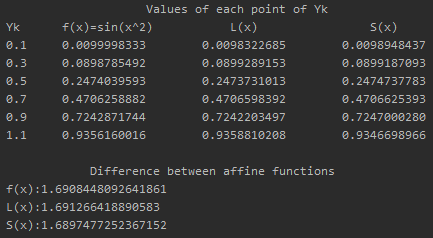


Отобразим каждую функцию по отдельности, не заметим разницы:

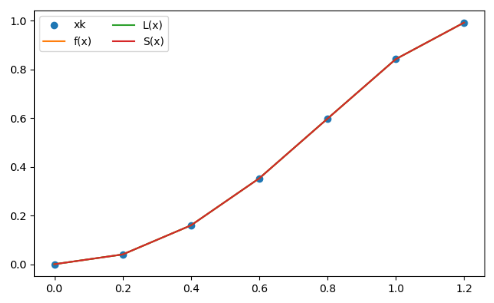


Можно сказать, что полином Лагранжа и сплайн-функция отлично аппроксимирует функцию вида sin(x2)

Если сравнить функции в узлах интерполирования, то конечно же во всех функциях значения будут истинными, а вот в заданных точках исследования(yk) мы можем увидеть различия, обратим внимания на данный рисунок, говорящий о том, какая из функций ближе приблизилась к истинному значению функции:



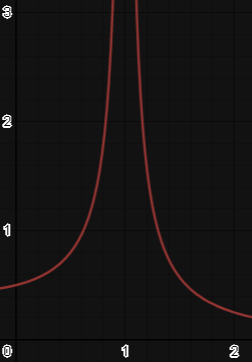
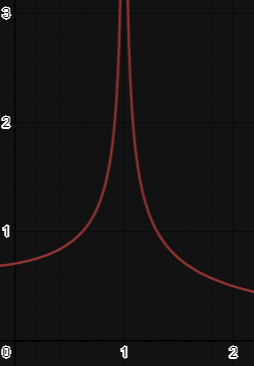
Разницу между функциями я решил показать, посчитав сумму квадратов вычисляемых по заданию точек (yk), в виде формулы это выглядит так: , чтобы было более наглядно отследить отклонения, и как мы можем увидеть от исходной функции L(x) оказался чуть выше, а S(x) оказался чуть ниже исходной функций, если бы это хотели увидеть на графике, но эти различия незначительны и на общем графике, при данном масштабе не видны, и на самом деле не существенны при не высоких требованиях к точности:



Перейдем ко второй части выполнения нашего задания.

**QUANC8.**

В начале оценим и поймем с каким видом подынтегральной функции мы имеем дело, отобразим ее графически на нашем промежутке и увидим точку разрыва:

(график при m = -1) (график при m = -0.5)

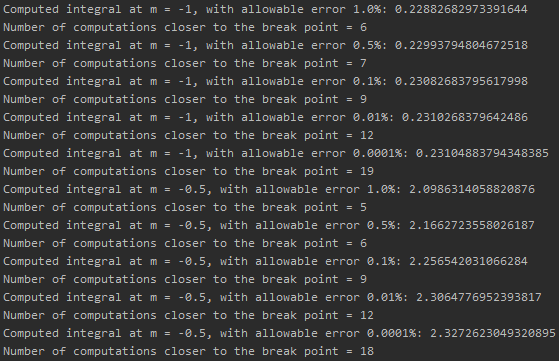
По рисунку можно увидеть, что точка разрыва нашей функции находится в точке x = 1, и от изменения степени m, конечно же не зависит, но из различия степени можно наблюдать более резкое сужение данной перевернутой впадины V-вида к точке разрыва при m = -0.5.

Посмотрим, как программа QUANC8 справиться с этим.



Зная, что внутри она использует адаптивную квадратурную формулу и то, что в ней есть ограничение на приближения(вплоть до сокращения промежутка до 30 раз) в тех промежутках, в которых хотелось бы получить как можно более точное значение, она отработает как надо. Проведем вычисления интеграла на двух промежутках (0, 1) и (1, 2) с различным приближением к правому и левому краю соответственно, т.е. к точке разрыва: ±1% , ±0.5%, ±0.1% , ±0.01% и ±0.0001%.

В дополнении, после полученного значения интеграла, выведем на экран понадобившееся количество делений промежутка при приближении к точке разрыва в момент работы программы.



**Выводы:**

Из результатов работы и вычисления значений в точках для всех трех функций, можно сделать вывод о том, что сплайн-интерполяция показывает худшие результаты, по сравнению с полиномом Лагранжа на краях промежутка. В точках внутри промежутка точность интерполяции полиномом Лагранжа и сплайнами примерно одинакова. Разница ровным счётом зависит конкретно от того с каким видом функции мы имеем дело, и результат работы и значений, полученных после построения L(x), S(x) зависит от того, как они работают изнутри.

В работе над второй частью задания, можно с точностью сказать, что QUANC8 отлично отрабатывает и с функцией с точкой разрыва, с требуемой погрешностью и точностью, которую нам надо получить. Исходя из полученных результатов, и взяв наибольшее требование к точности можно понять что значение интеграла на промежутке [0, 2] при m = -1 приблизительно равно 0.231,такое значение мы можем наблюдать уже с допустимой ошибкой в 0.01%, в тот же момент при m= -0.5 значение приблизительно равно 0.1323, что может наблюдать уже даже при ошибке в 0.5% поэтому в дальнейшем с помощью такой оценки можем воспользоваться этим и сэкономить на вычислительной мощности и времени расчета, если это нужно, например для более сложных задач, если требования к точности достаточно не высокое.

Смотря и анализируя полученное количество вычислений подынтегральной функции для различных степеней приближения к точке разрыва удостоверились на практике, что чем ближе будут границы интегрирования к точке разрыва, тем больше понадобится итераций.

Такая разница между двумя функция при m = -1, -0.5 могла быть замечена еще при первом рассмотрении (на рисунках), на которых как раз таки мы и обозначали что более резкий рост к точке разрыва происходит у функции при m = -0.5, что и означает, что чем ближе к точке разрыва мы находимся тем меньше объема покрываемой области под функцией мы можем записать в значение интеграла.

Весь исходный код и файлы можно найти в моем профиле:

[github.com/b0r1ngx/ComputationalMath](https://github.com/b0r1ngx/ComputationalMath)