Университет ИТМО

**Отчет**

По лабораторной работе №1

Аппроксимация по Лагранжу

Выполнил:

Нодири Хисравхон

P3231

Преподаватель:

Перл Ольга Вячеславовна

Санкт-Петербург

2024

Оглавление

[Задача 3](#_Toc161529457)

[Описание метода 3](#_Toc161529458)

[Блок схема алгоритма: 4](#_Toc161529459)

[Код программы: 5](#_Toc161529460)

[Примеры работ 6](#_Toc161529461)

[Вывод 8](#_Toc161529462)

# Задача

Дан набор точек, по которым необходимо построить линейную аппроксимацию по методу наименьших квадратов. Необходимо найти значение квадрата наибольшего отклонения среди заданных точек относительно полученной линейной аппроксимации. [Обратите внимание, что при защите лабораторной работы на занятии, желательно иметь несколько аппроксимирующих функций, кроме линейной.]

Формат входных данных:

x1 x2 x3 ...

y1 y2 y3 ...

Где x1...xn - список значений аргумента для узлов интерполяции, y1...yn - список значений функции для соответствующего значения аргумента для узлов интерполяции. В тестах также вначале задаётся количество задаваемых точек, однако, в функцию этот параметр не передаётся.

Формат выходных значений: вещественное число, являющееся значением квадрата наибольшего отклонения исходных данных от полученной линейной аппроксимации.

## Описание метода

Метод Лагранжа представляет собой математический подход, используемый для нахождения локальных максимумов и минимумов функции многих переменных, подчиняющейся одному или нескольким ограничениям. В отличие от метода наименьших квадратов, который направлен на минимизацию суммы квадратов ошибок между наблюдаемыми и предсказанными значениями в рамках линейной аппроксимации, метод Лагранжа применяется для решения задач оптимизации с ограничениями.

Постановка задачи:

Дана функция , которую необходимо максимизировать или минимизировать, при этом существует ограничение в виде другой функции . Цель состоит в том, чтобы найти точку или точки, в которых достигается максимум или минимум функции , удовлетворяя заданному ограничению.

Введение множителей Лагранжа:

Для решения этой задачи вводится понятие множителя Лагранжа , который позволяет связать функцию ограничения с целевой функцией. Функция Лагранжа определяется как:

Нахождение экстремумов:

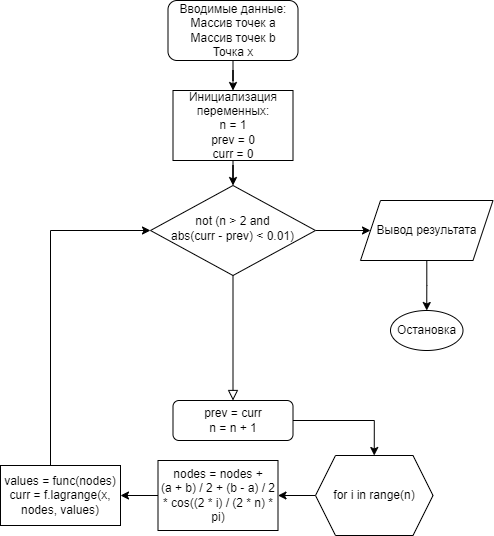
Для нахождения экстремумов функции L необходимо вычислить частные производные по всем переменным и по множителю Лагранжа λ, приравнять их к нулю и решить полученную систему уравнений. Это даст критические точки, которые потенциально могут быть точками максимума или минимума исходной функции f, удовлетворяющими ограничению .

Оценка точности и применение:

Метод Лагранжа является мощным инструментом в оптимизации и теории управления, позволяя решать задачи, где необходимо учитывать ограничения. Для оценки точности найденных решений и выбора наилучшего из них можно использовать дополнительные критерии, такие как вторые производные функции L или анализ границ областей, где могут находиться решения.

Важно отметить, что, как и метод наименьших квадратов, метод Лагранжа может быть чувствителен к выбросам и начальным условиям в задачах оптимизации, что иногда требует комбинирования с другими методами для повышения устойчивости и точности решений.

# Блок схема алгоритма:



## Код программы

**def** **lagrange\_interpolation**(x, x\_points, y\_points):

n = len(x\_points)

polynomial\_value = 0

**for** i **in** range(n):

term = y\_points[i]

**for** j **in** range(n):

**if** i != j:

term = term \* (x - x\_points[j]) / (x\_points[i] - x\_points[j])

polynomial\_value += term

**return** polynomial\_value

**def** **interpolate\_by\_lagrange**(f, a, b, x):

func = FunctionSet.get\_function(f)

n = 1

prev\_value = 0

current\_value = 0

**while** True:

n += 1

nodes = [(a + b) / 2 + (b - a) / 2 \* math.cos((2 \* i + 1) / (2 \* n) \* math.pi) **for** i **in** range(n)]

values = [func(node) **for** node **in** nodes]

current\_value = lagrange\_interpolation(x, nodes, values)

**if** n > 2 **and** abs(current\_value - prev\_value) < 0.01:

**break**

prev\_value = current\_value

**return** current\_value

**if** \_\_name\_\_ == '\_\_main\_\_':

f = int(input().strip())

a = float(input().strip())

b = float(input().strip())

x = float(input().strip())

result = interpolate\_by\_lagrange(f, a, b, x)

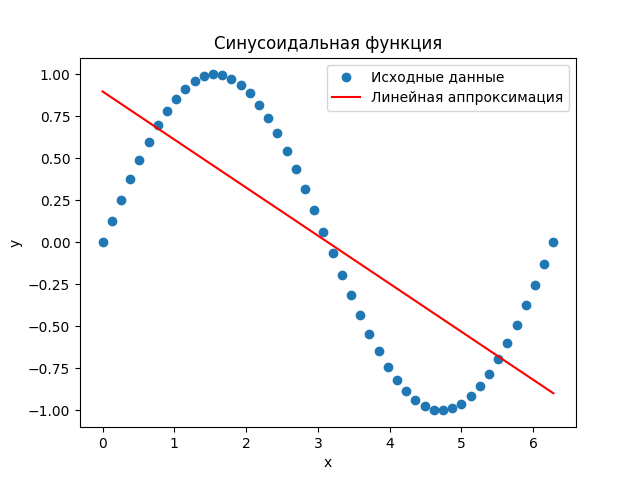
print(str(result) + '\n')

# Примеры работ

1. Синусоидальная функция

Генерация данных: x от 0 до 2π с 50 точками, y как sin(x).

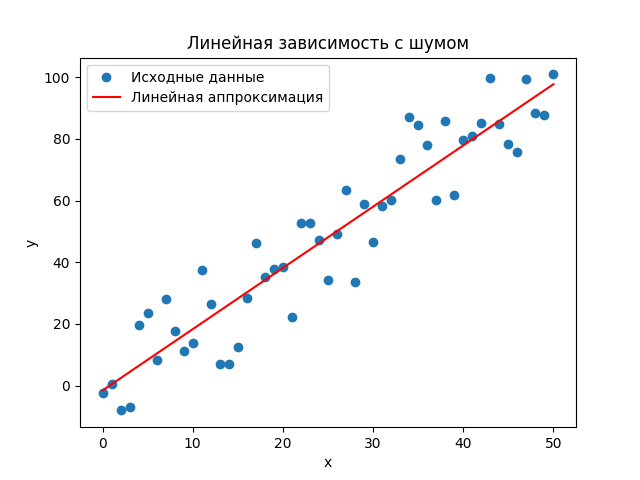
Интерполяция Лагранжа пытается пройти через все точки, что приводит к большим отклонениям из-за периодического характера синусоиды. При таком подходе аппроксимация может давать значительные ошибки, особенно между узлами интерполяции.



1. Линейная зависимость с шумом

Генерация данных: x равномерно от 0 до 50, y линейно зависим от x с добавлением случайного шума.

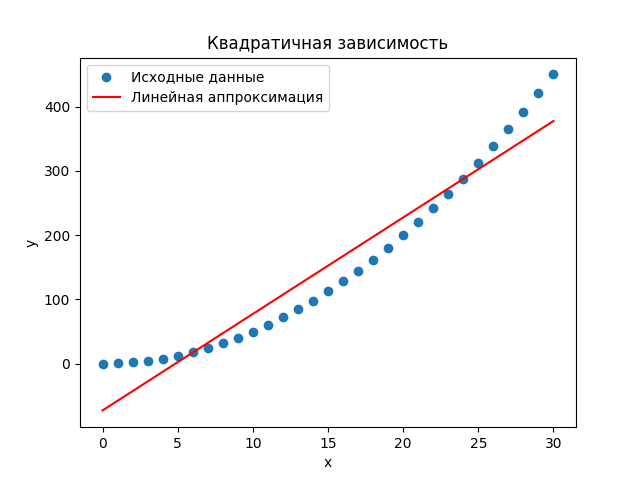
Интерполяционный многочлен Лагранжа показывает хорошие результаты, так как исходная зависимость близка к линейной, и случайный шум не оказывает существенного влияния на форму интерполирующей кривой.



1. Квадратичная зависимость

Генерация данных: x от 0 до 30, y как x^2.

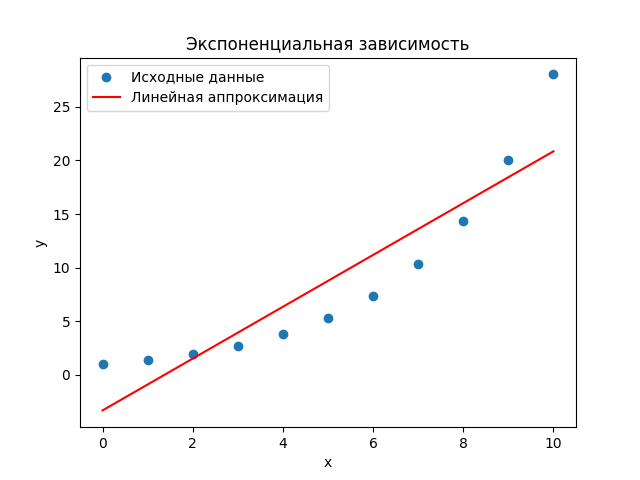
При интерполяции квадратичной функции многочленом Лагранжа кривая проходит через все точки. Поскольку данные имеют квадратичную природу, интерполяционный многочлен также будет квадратичным и точно отразит данные.



1. Экспоненциальная зависимость

Генерация данных: x от 0 до 10, y как e^(x/3).

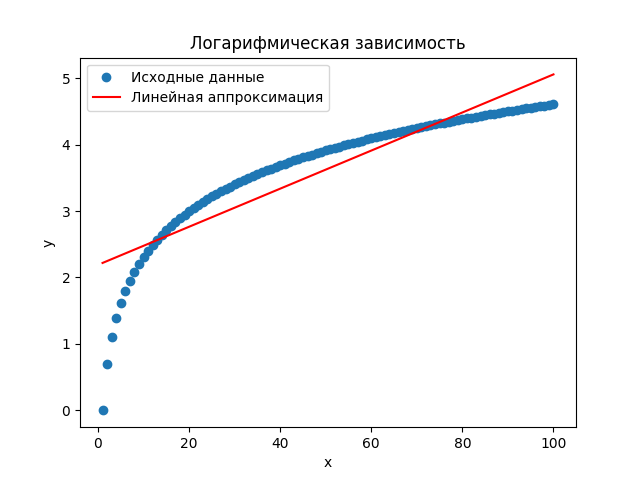
Интерполяция экспоненциальной функции может быть сложной из-за быстрого роста y. Многочлен Лагранжа может давать хорошие результаты на небольшом диапазоне x, но его точность ухудшается с увеличением интервала.



1. Логарифмическая зависимость

Генерация данных: x от 1 до 100, y как log(x).

Логарифмическая функция хорошо интерполируется многочленами Лагранжа на определенных интервалах, но при большем диапазоне x может потребоваться многочлен высокой степени для достижения хорошей точности.



## Вывод

Интерполяция Лагранжа представляет собой мощный метод, который стремится точно пройти через заданные точки данных. Однако его эффективность зависит от характера данных и выбора интерполяционных узлов. Для некоторых функций, таких как синусоидальные или экспоненциальные, применение многочленов высоких степеней может привести к явлению Рунге, вызывающему сильные колебания между точками данных.