Министерство науки и высшего образования

Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования

«Московский государственный технический университет имени Н.Э. Баумана

(национальный исследовательский университет)»

(МГТУ им. Н.Э. Баумана)

ФАКУЛЬТЕТ «Информатика и системы управления»

КАФЕДРА «Программное обеспечение ЭВМ и информационные технологии (ИУ-7)»

Лабораторная работа № 4

Тема: Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

Студент Сироткина П.Ю.

Группа ИУ7-46Б

Оценка (баллы)

Преподаватель Градов В.М.

Москва, 2021 год

Цель работа:

Получение навыков построения алгоритма метода наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами.

Исходные данные:

1. Таблица функции с весами p_i с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек.
2. Степень аппроксимирующего полинома - n.

Результат: аппроксимирующий полином. Графическая иллюстрация: полином и сами точки.

Описание алгоритма: TODO

Код программы:

Решение СЛАУ для нахождения коэффициентов полинома:

```
def solve_slae(p, x, y, degree):

    slae = make_slae(p, x, y, degree)

    for i in range(degree + 1):
        for j in range(degree + 1):
            if i == j:
                continue
            mul = slae[j][i] / slae[i][i]
            for k in range(degree + 2):
                slae[j][k] -= mul * slae[i][k]

    for i in range(degree + 1):
        div = slae[i][i]
        for j in range(degree + 2):
            slae[i][j] /= div

    return [slae[i][-1] for i in range(len(slae))]
```

Подфункция составления СЛАУ по имеющимся данным:

```
def make_slae(p, x, y, degree):

    slae = [0] * (degree + 1)
    for i in range(degree + 1):
        slae[i] = [0] * (degree + 2)

    for i in range(degree + 1):
        for j in range(degree + 1):
            slae[i][j] = get_coeff(p, x, x, i, j)
        slae[i][degree + 1] = get_coeff(p, y, x, 1, i)

    return slae
```

Подфункция вычисления так называемого скалярного произведения:

```python
def get_coeff(p, a, b, degree_a, degree_b):
    coeff = 0

    for i in range(len(p)):
        coeff += p[i] * (a[i] ** degree_a) * (b[i] ** degree_b)

    return coeff
```

Примечание:

В отчете приведены только ключевые функции. Организация меню, ввода и редактирования данных, рисовка графиков опущены, т.к. мешают описанию и понимаю главной идеи алгоритма в отчете.

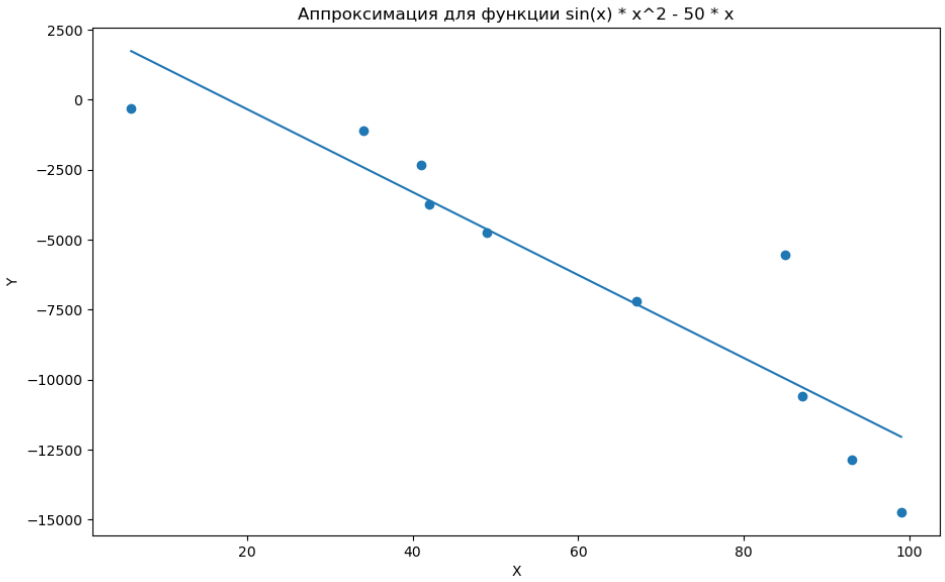
Полный код программы можно посмотреть в приложении к отчету.

Результат работы программы:

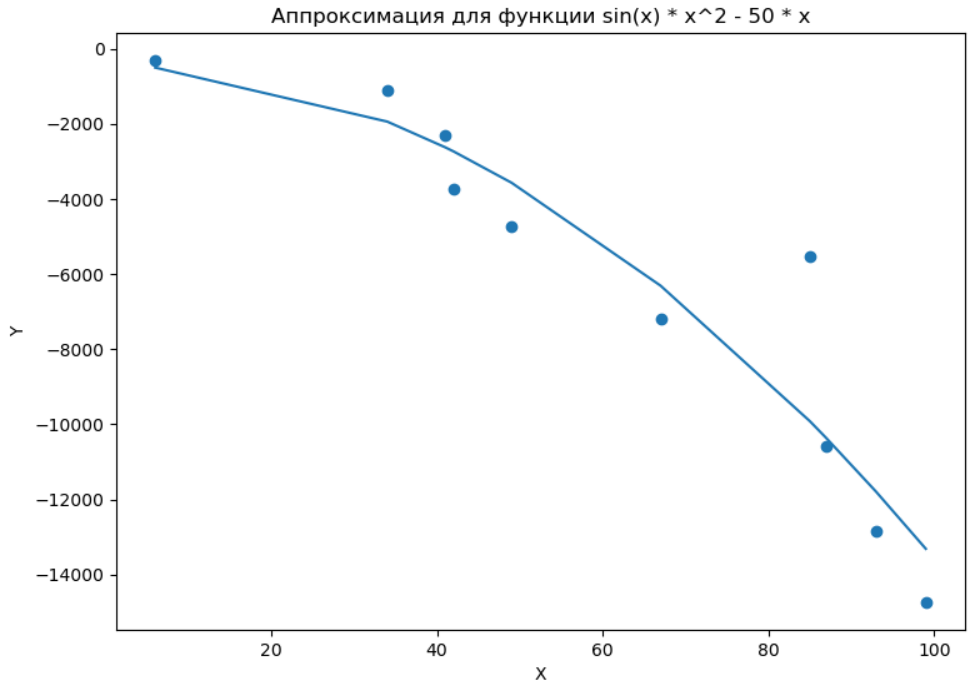
1. Все веса точек равны 1. Рабочая таблица:

№ узла	X	Y	Вес
1	6.00	-310.06	1.00
2	34.00	-1088.38	1.00
3	41.00	-2316.64	1.00
4	42.00	-3716.74	1.00
5	49.00	-4739.96	1.00
6	67.00	-7190.43	1.00
7	85.00	-5522.15	1.00
8	87.00	-10570.34	1.00
9	93.00	-12851.69	1.00
10	99.00	-14743.23	1.00

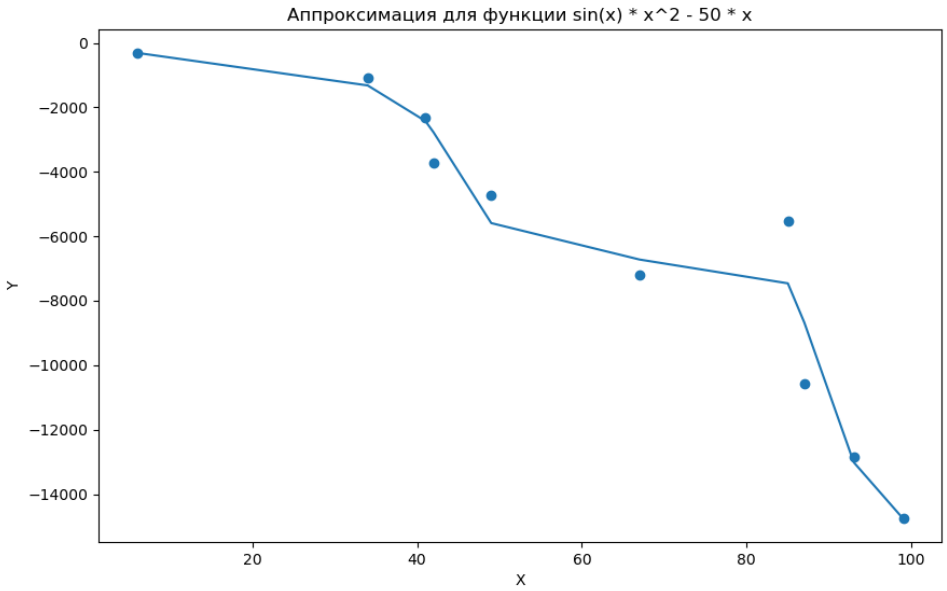
Результат при n = 1:



Результат при n = 2:

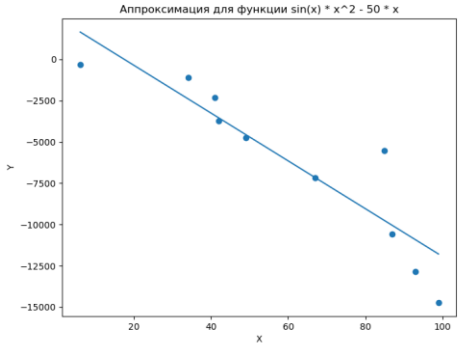


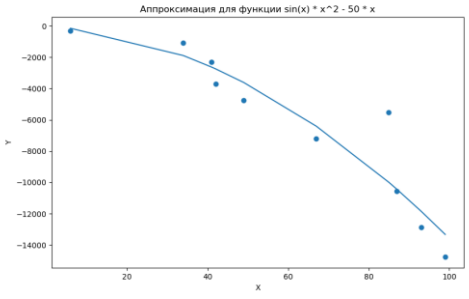
Результат при n = 6:

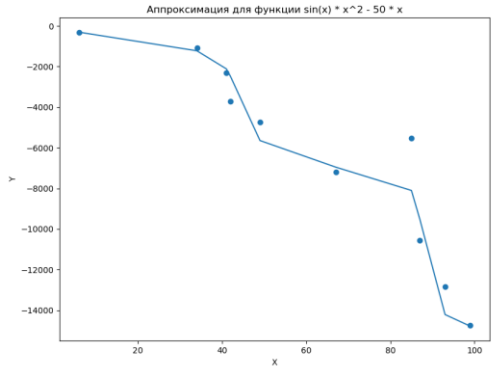


2. Случай, когда веса точек различны. Рабочая таблица:

№ узла	X	Y	Вес
1	6.00	-310.06	0.51
2	34.00	-1088.38	0.78
3	41.00	-2316.64	0.32
4	42.00	-3716.74	0.24
5	49.00	-4739.96	0.40
6	67.00	-7190.43	0.83
7	85.00	-5522.15	0.30
8	87.00	-10570.34	0.68
9	93.00	-12851.69	0.03
10	99.00	-14743.23	0.36

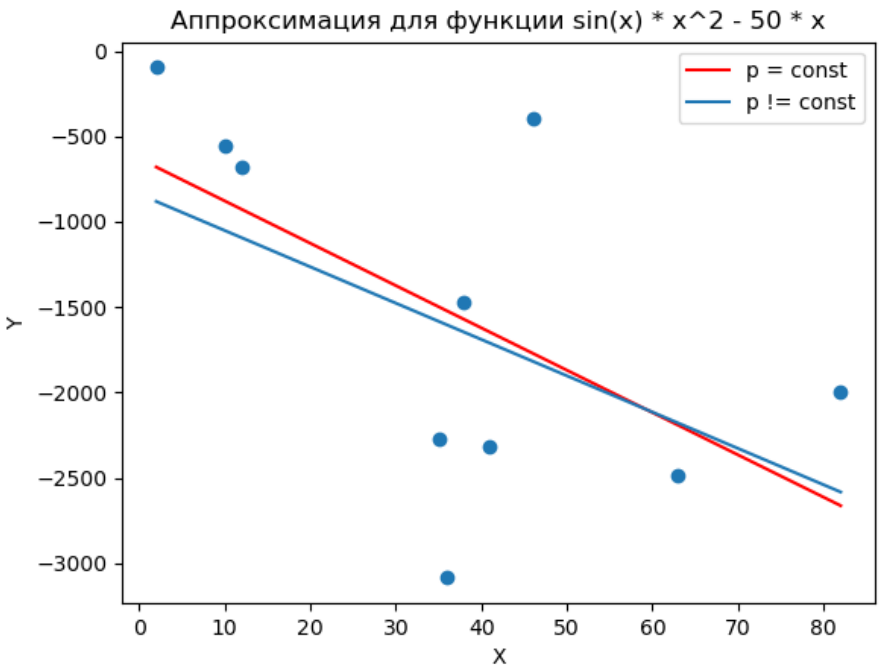




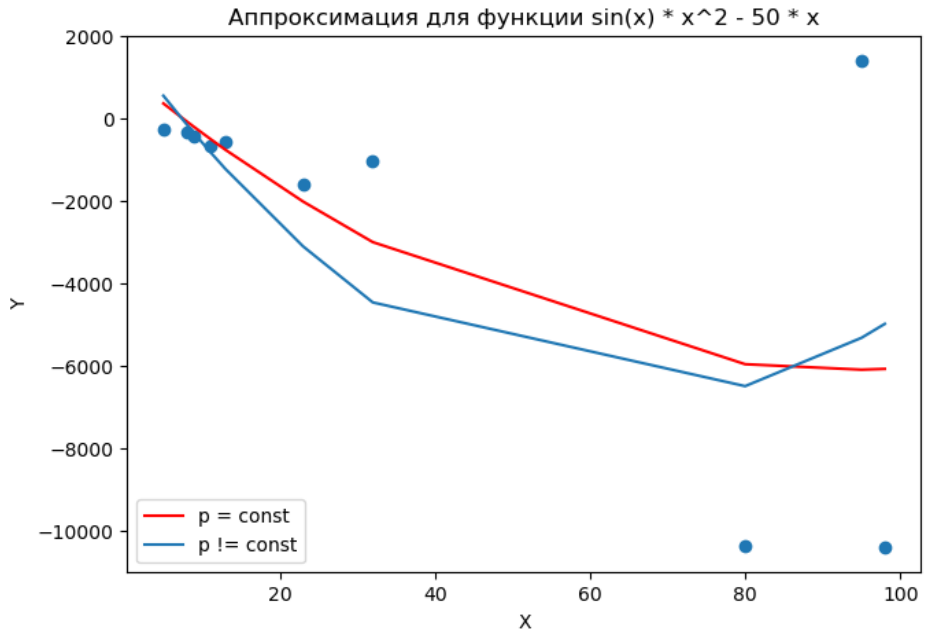


Сравнение:

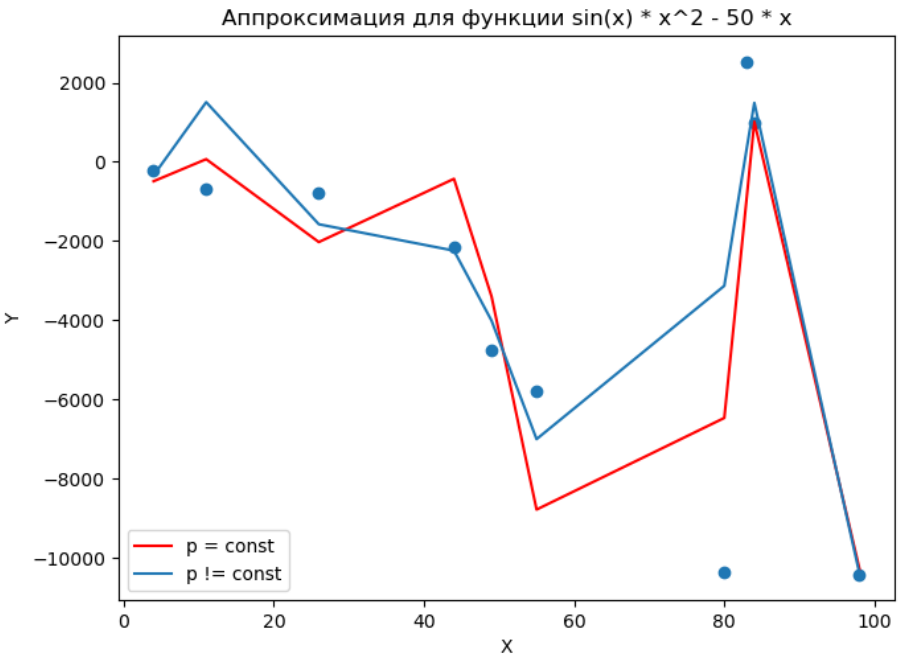
n = 1


Аппроксимация для функции sin(x) * x^2 - 50 * x
p = const
p != const
0
−500
−1000
−1500
−2000
−2500
−3000
Y
0
10
20
30
40
50
60
70
80
X

n = 2


Аппроксимация для функции sin(x) * x^2 - 50 * x
2000
0
−2000
−4000
−6000
−8000
−10000
Y
p = const
p != const
20
40
60
80
100
X

n = 6



Ответы на контрольные вопросы: