МГТУ им. Н. Э. Баумана

Отчет по лабораторной работе №4 по предмету «Вычислительные алгоритмы»
«Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения»

Выполнила Параскун София,
ИУ7-44Б
Проверил
Градов В. М.

Москва, 2021 г.

Цель работы: получение навыков построения алгоритма методами наименьших квадратов с использованием полинома заданной степени при аппроксимации табличных функций с весами.

1. Исходные данные

1. Таблица функции с весами p_i с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек.

x	y	ρ_i

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

2. Степень аппроксимирующего полинома — n.

2. Код программы

Программа состоит из файла main.py. Таблица функции генерируется внутри программы.

```
import random
import numpy as np
import matplotlib.pyplot as plt
from math import sin

class Coefficient():
   def __init__(self):
      self.xCoef = []
      self.yCoef = []

   def primeCoefCulc(self, data):
      for i in range(data.n + 1):
         self.xCoef.append([])

      for i in range(data.n + 1):
         for j in range(i, data.n + 1):
            curCoef = 0
            for k in range(data.count):
               curCoef += data.p[k] * pow(data.x[k], i + j)
            self.xCoef[i].append(curCoef)
            if i < j:
               self.xCoef[j].append(curCoef)
```

```
      for i in range(data.n + 1):
         curCoef = 0
         for j in range(data.count):
            curCoef += data.p[j] * data.y[j] * pow(data.x[j], i)
         self.yCoef.append(curCoef)

      self.npXCoef = np.asarray(self.xCoef)
      self.npYCoef = np.asarray(self.yCoef)

      self.npResCoef = np.linalg.solve(self.npXCoef, self.npYCoef)

      # for i in range(len(self.npResCoef)):
      #      print("a[{0:3d}] = {1:6.3f}".format(i, self.npResCoef[i]))

class Data():
   def __init__(self):
      self.count = 20
      self.x = []
      self.y = []
      self.p = []
      self.n = 0
      self.nBuf = 0

      for i in range(self.count):
         self.x.append(random.uniform(0, 10))
      self.x.sort()

      for i in range(self.count):
         self.y.append(self.x[i] + sin(self.x[i]))
         self.p.append(random.uniform(0.01, 20))

   def inputAll(self):
      self.__init__()
      self.output()
      print("Do you want to choose weights? [y/1/n] ", end = '')
      changeF = input()
      if changeF == 'y':
         self.weightsChange()
         self.output()
      elif changeF == '1':
         for i in range(self.count):
            self.p[i] = 1
         self.output()
      self.inputN()

   def inputN(self):
      print("Input n: ", end = '')
      self.n = int(input())
```

```
def output(self):
   print(" ind |      x     |       y    |       p")
   print('-' * 44)
   for i in range(self.count):
      print("{0:4d} | {1:10.3f} | {2:10.3f} | {3:10.3f}".format(i, self.x[i], self.y[i], self.p[i]))

def weightsChange(self):
   ind = 0
   while (ind != -1):
      print("Input index and value: ", end = '')
      ind = int(input())
      if ind != -1:
         value = float(input())
         self.p[ind] = value
   print("End of changes!")

def drawFunc(self, coef, mode):
   self.plotRealFunc()
   if mode == 1:
      print("mode = 1")
      self.nBuf = self.n
      self.n = 1
      coef.__init__()
      coef.primeCoefCulc(data)
      self.plotFunc(coef, mode)
      mode = 0
      print("mode = 0")
      self.n = 2
      coef.__init__()
      coef.primeCoefCulc(data)
      self.plotFunc(coef, mode)
      plt.plot(self.x, self.y, 'o')
      plt.xlabel("X")
      plt.ylabel("Y")
      plt.grid()
      plt.title("Наилучшее среднеквадртичное приближение")
      plt.legend()
      plt.show()
   else:
      print("mode = 2")
      self.n = self.nBuf
      coef.__init__()
      coef.primeCoefCulc(data)
      self.plotFunc(coef, mode)
      mode = 3
      print("mode = 3")
      for i in range(self.count):
         self.p[i] = 1
```

```
        coef.__init__()
        coef.primeCoefCulc(data)
        self.plotFunc(coef, mode)
        plt.plot(self.x, self.y, 'o')

        plt.xlabel("X")
        plt.ylabel("Y")
        plt.grid()
        plt.title("Наилучшее среднеквадртичное приближение")
        plt.legend()
        plt.show()

    def plotRealFunc(self):
        xBuf = np.linspace(self.x[0], self.x[self.count - 1], 1000)
        yBuf = []
        for i in range(len(xBuf)):
            yBuf.append(xBuf[i] + sin(xBuf[i]))
        plt.plot(xBuf, yBuf, color = "m", label = "Исходная функция")

    def plotFunc(self, coef, mode):
        xBuf = np.linspace(self.x[0], self.x[self.count - 1], 1000)
        yBuf = []
        for i in range(len(xBuf)):
            yCur = coef.npResCoef[0]
            for j in range(1, self.n + 1):
                yCur += coef.npResCoef[j] * pow(xBuf[i], j)
            yBuf.append(yCur)

        if mode == 0:
            plt.plot(xBuf, yBuf, color = "r", label = "Равные весы (n={0})".format(self.n))
        elif mode == 1:
            plt.plot(xBuf, yBuf, color = "g", label = "Равные веса (n={0})".format(self.n))
        elif mode == 2:
            plt.plot(xBuf, yBuf, color = "b", label = "Разные веса (n={0})".format(self.n))
        else:
            plt.plot(xBuf, yBuf, color = "y", label = "Веса равны 1 (n={0})".format(self.n))

if __name__ == "__main__":
    data = Data()
    data.inputAll()
    coef = Coefficient()
    coef.__init__()
    coef.primeCoefCulc(data)

    data.drawFunc(coef, 1)
    data.drawFunc(coef, 2)
```

3. Результаты работы

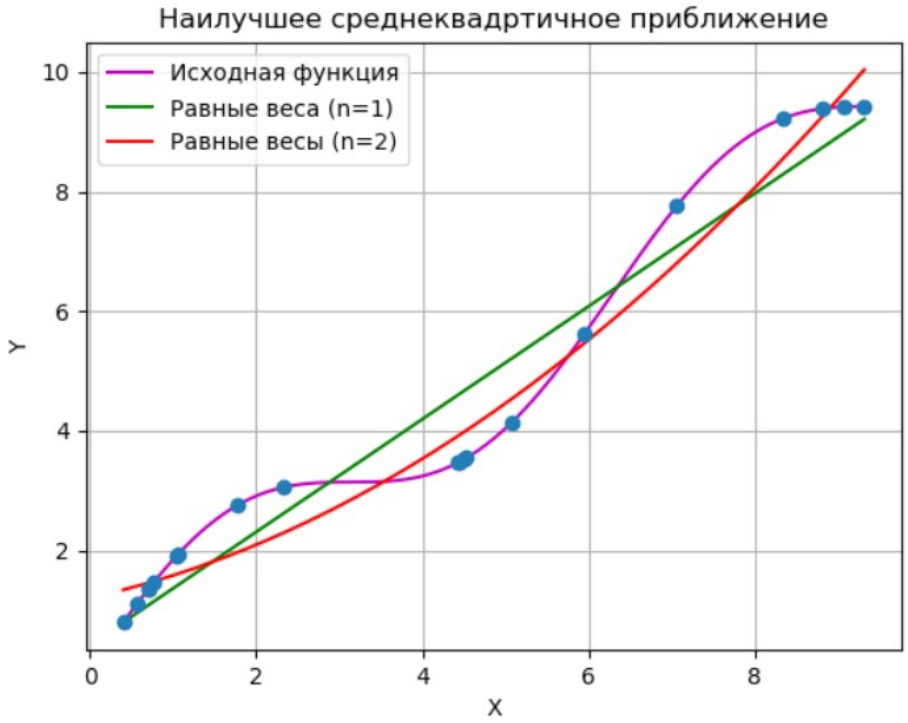
Графики, построенные по аналогии с рис. 1 в тексте лекции №4: точки — заданная табличная функция, кривые — найденные полиномы.
Обязательно приводить таблицы, по которым работала программа.

Выбранная функция y = x + sin(x). Значения аргументов x и значения весов p генерируются случайным образом.

1. Веса всех точек равны 1. Построить полином степеней n=1 и 2. Можно привести результаты и при других степенях полинома, однако, не загроможденная сильно при этом рисунок.

Для сгенерированной таблицы графики функции будут следующими:

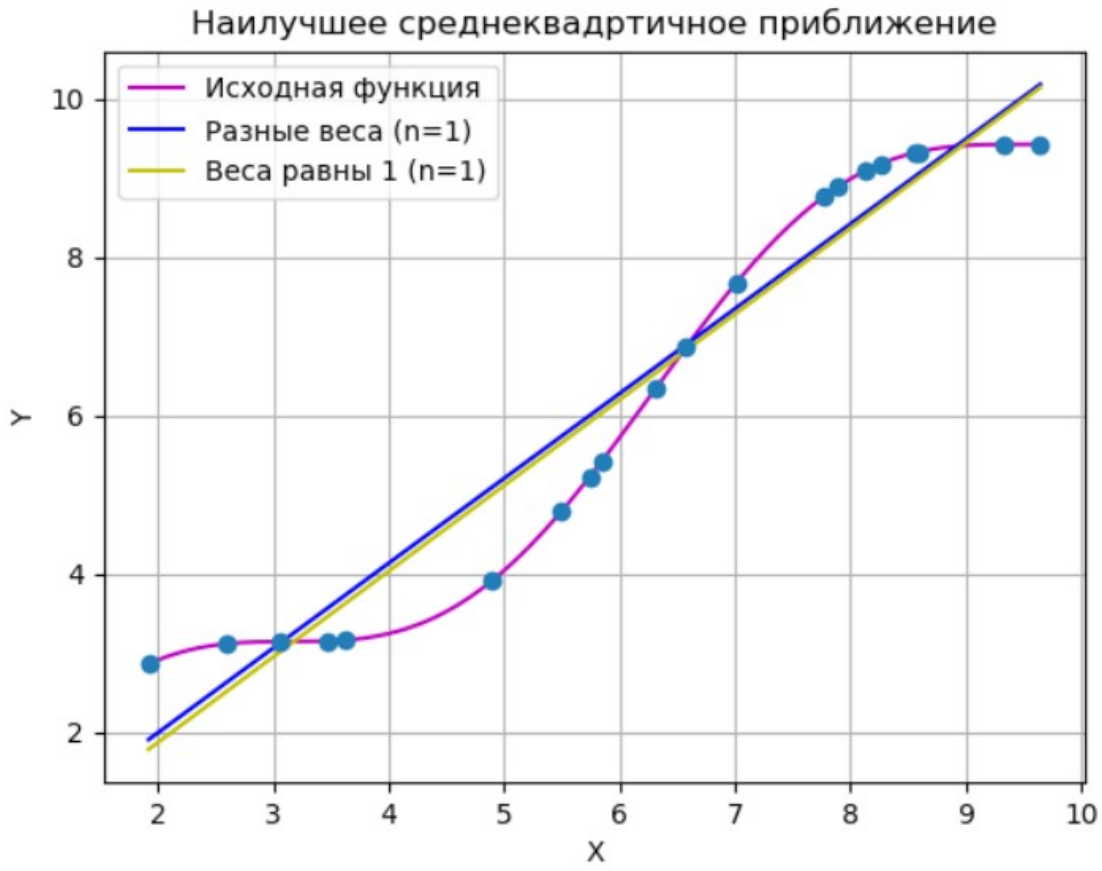
ind	x	y	p
0	0.404	0.796	1.000
1	0.567	1.104	1.000
2	0.700	1.344	1.000
3	0.710	1.361	1.000
4	0.765	1.458	1.000
5	1.040	1.902	1.000
6	1.070	1.947	1.000
7	1.767	2.748	1.000
8	2.339	3.058	1.000
9	4.427	3.468	1.000
10	4.450	3.484	1.000
11	4.509	3.530	1.000
12	4.528	3.545	1.000
13	5.074	4.139	1.000
14	5.953	5.629	1.000
15	7.062	7.765	1.000
16	8.329	9.218	1.000
17	8.809	9.387	1.000
18	9.060	9.417	1.000
19	9.315	9.425	1.000



Как видим, функция первой степени представляет собой прямую, функция второй степени уже имеет изгиб.

2. Веса точек разные. Продемонстрировать, как за счет назначения весов точкам можно изменить положение на плоскости прямой линии (полином первой степени), аппроксимирующей один и тот же набор точек (одну таблицу y(x)). Например, назначая веса узлам в таблице изменить знак углового коэффициента прямой. На графике в итоге должны быть представлены точки исходной функции и две аппроксимирующие их прямые линии. Одна отвечает значениям $p_i=1$, а другая — назначенным разными весам точек. Информацию о том, какие веса были использованы в расчете обязательно указать, чтобы можно было проконтролировать работу программы (лучше это сделать в виде таблицы).

ind	x	y	p
0	1.920	2.860	16.771
1	2.592	3.114	8.240
2	3.057	3.141	9.136
3	3.469	3.147	10.895
4	3.627	3.160	7.146
5	4.893	3.909	11.080
6	5.498	4.791	19.138
7	5.740	5.223	1.833
8	5.845	5.421	3.062
9	6.308	6.334	9.303
10	6.575	6.863	0.576
11	7.012	7.679	7.973
12	7.773	8.770	9.001
13	7.891	8.891	12.997
14	8.136	9.096	15.805
15	8.262	9.180	13.890
16	8.565	9.323	15.410
17	8.584	9.329	8.213
18	9.335	9.425	3.392
19	9.645	9.427	13.450



Как видим, для графика с равными весами угол наклона больше, однако это может быть обусловлено видом функции (y = x + sin(x)).

4. Вопросы при защите лабораторной работы

1. Что произойдет при задании степени полинома n=N-1 (число узлов таблицы минус 1)?

Для однозначного определения полинома N-1 степени достаточно N точек, что означает, что полином будет построен таким образом, что его график будет проходить через все табличные точки. При такой конфигурации в выражении

$$\sum_{i=1}^{N} \rho_i [y(x_i) - \varphi(x_i)]^2 = min$$

часть выражения, находящаяся в скобках, обратится в 0, что означает, что нет зависимости от весов (при любых заданных весах, значение полинома будет минимальным в случае прохода через табличные точки).

2. Будет ли работать Ваша программа при n>=N? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?

Программа работать будет, но ее работа будет некорректна, так как начиная со случая n=N определить СЛАУ, которую необходимо решить, будет тождественно равен 0 (уравнения СЛАУ не будут линейно-независимыми). Аварийная остановка программы может произойти при приведении диагональной матрицы к единичной (СЛАУ решалось методом Гаусса-Жордана, поэтому предполагается приведение диагональной матрицы к единичной), где может произойти деление на 0. Анализ можно проводить при решении СЛАУ или же на начальном этапе (ввод степени полинома).

3. Получить формулу для коэффициента полинома a_0 при степени полинома n=0. Какой смысл имеет величина, которую представляет данный коэффициент?

Полученная формула

$$a_0 = \frac{\sum_{i=1}^{N} \rho_i y_i}{\sum_{i=1}^{N} \rho_i}$$

где p_i – все I точки. Данную формулу можно преобразовать делением числителя и знаменателя на сумму весов, из чего получится математическое ожидание

$$M[X] = \sum_{i=1}^{\infty} x_i \rho_i$$

4. Записать и вычислить матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N-2. Принять все $p_i = 1$.

Пусть таблица точек имеет вид

x_i	y_i	ρ_i
x_0	y_0	1
x_1	y_1	1

Тогда СЛАУ примет вид

$$\begin{cases} a_0 + (x_0 + x_1)a_1 + (x_0^2 + x_1^2)a_2 = y_0 + y_1 \\ (x_0 + x_1)a_0 + (x_0^2 + x_1^2)a_1 + (x_0^3 + x_1^3)a_2 = y_0x_0 + y_1x_1 \\ (x_0^2 + x_1^2)a_0 + (x_0^3 + x_1^3)a_1 + (x_0^4 + x_1^4)a_2 = y_0x_0^2 + y_1x_0^2 \end{cases}$$

$$\Delta = (x_0^2 + x_1^2)(x_0^4 + x_1^4) + (x_0 + x_1)(x_0^3 + x_1^3)(x_0^2 + x_1^2) + (x_0^2 + x_1^2)(x_0 + x_1)(x_0^3 + x_1^3) - (x_0^2 + x_1^2)(x_0^2 + x_1^2)(x_0^2 + x_1^2) - (x_0^3 + x_1^3)(x_0^3 + x_1^3) - (x_0 + x_1)(x_0 + x_1)(x_0^4 + x_1^4) = 0$$

Так как Δ = 0, система решений не имеет, что было упомянуто в ответ на вопрос 2.

5. Построить СЛАУ при выборочном задании степеней аргумента полинома f(x) = $a_0 + a_1x^m + a_2x^n$, причем степени n и m в этой формуле известны.

$$\begin{cases} (x^0, x^0)a_0 + (x^0, x^m)a_1 + (x^0, x^n)a_2 = (y, x^0) \\ (x^m, x^0)a_0 + (x^m, x^m)a_1 + (x^m, x^n)a_2 = (y, x^m) \\ (x^n, x^0)a_0 + (x^n, x^m)a_1 + (x^n, x^n)a_2 = (y, x^n) \end{cases}$$

6. Предложить схему алгоритма решения задачи из вопроса 5, если степени n и m подлежат определению наравне с коэффициентами ak, m. е. количество неизвестных равно 5.

1. Перебираем все возможные пары n и m .
2. Для каждой пары ищем все коэффициенты a , а также ошибку.
3. Среди всех таких наборов выбираем тот, у которого ошибка будет наименьшей