|  |  |
| --- | --- |
| Gerb-BMSTU_01 | **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\_\_\_\_\_\_\_\_

**Лабораторная работа № 4**

**Тема** Построение и программная реализация алгоритма наилучшего среднеквадратичного приближения.

**Студент** Серова М. Н.

**Группа** ИУ7-45Б

**Оценка (баллы)** \_\_\_\_\_\_\_\_

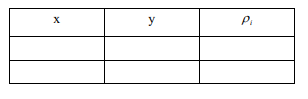
**Преподаватель** Градов В. М.

*Москва,*

*2021 г.*

**Цель работы.** 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 ρi с количеством узлов N. Сформировать таблицу самостоятельно со случайным разбросом точек. 

Предусмотреть в интерфейсе удобную возможность изменения пользователем весов в таблице.

1. Степень аппроксимирующего полинома – n.

**Код программы**

Код программы представлен на листинге 1

|  |
| --- |
| Листинг 1. square.py  import numpy as np  from tkinter import Frame, Tk, Entry, Button, Label, N  from tkinter import ttk  import tkinter.messagebox as box  import matplotlib  matplotlib.use('TkAgg')  from matplotlib.backends.backend\_tkagg import FigureCanvasTkAgg  from matplotlib.figure import Figure  class **SquareApproximation**(**Frame**):      def **\_\_init\_\_**(self, parent):          super().**\_\_init\_\_**(parent)          self.parent = parent          self.parent.title("Среднеквадратическое приближение")          self.init\_ui()      def **fill\_points**(self):          self.x = np.random.randint(-100, 100, self.N)          self.y = np.random.randint(-100, 100, self.N)          self.ro = np.ones(self.N)      def **enter\_amount**(self):          try:              N = int(self.amount\_entry.get())          except ValueError:              box.showerror("Ошибка", "Количество точек должно быть целым числом")              return          if (N <=0 or N > 20):              box.showerror("Ошибка", "Количество точек должно быть больше нуля и меньше 20")              return          self.N = N          self.fill\_points()          x\_label = Label(self, text="X")          x\_label.grid(row=3, column=0)          y\_label = Label(self, text="Y")          y\_label.grid(row=3, column=1)          ro\_label = Label(self, text="P")          ro\_label.grid(row=3, column=2)          self.x\_entry\_list = []          self.y\_entry\_list = []          self.ro\_entry\_list = []            for i in **range**(self.N):              self.x\_entry\_list.append(Entry(self))              self.x\_entry\_list[i].insert(0, self.x[i])              self.x\_entry\_list[i].configure(width=20)              self.x\_entry\_list[i].grid(row= 4 + i, column=0)                self.y\_entry\_list.append(Entry(self))              self.y\_entry\_list[i].insert(0, self.y[i])              self.y\_entry\_list[i].configure(width=20)              self.y\_entry\_list[i].grid(row = 4 + i, column=1)              self.ro\_entry\_list.append(Entry(self))              self.ro\_entry\_list[i].insert(0, self.ro[i])              self.ro\_entry\_list[i].configure(width=20)              self.ro\_entry\_list[i].grid(row = 4 + i, column=2)          startButton = Button(self, text="Найти полиномы", command=self.get\_polynoms)          startButton.grid(row = 4 + self.N, column=2)      def **get\_polynoms**(self):          x\_list = []          y\_list = []          ro\_list = []          try:              for i in **range**(self.N):                  x\_list.append(float(self.x\_entry\_list[i].get()))                  y\_list.append(float(self.y\_entry\_list[i].get()))                  ro\_list.append(float(self.ro\_entry\_list[i].get()))          except ValueError:              box.showerror("Ошибка", "Координаты точки и вес должны быть вещественными числами")              return          self.x = np.array(x\_list)          self.y = np.array(y\_list)          self.ro = np.array(ro\_list)          mode = self.mode.get()          if (mode == "Изменение степени полинома"):              self.get\_polynoms\_change\_pow()          else:              self.get\_polynoms\_change\_weight()      def **get\_polynoms\_change\_weight**(self):          ro = np.ones(self.N)            if (self.drawed):              self.f.clear()          else:              self.drawed = True          self.f = Figure(figsize=(10, 5), dpi=93)          ax = self.f.add\_subplot(111)          n = 1          res = self.get\_res\_coeffs(n, ro)          ax.plot(self.x, self.y, 'x', color="r", label="Исходные точки")          y = []          x = []          step = (**max**(self.x) - **min**(self.y)) / 100          i = **min**(self.x)          while (i <= **max**(self.x)):              x.append(i)              y.append(res[0] + res[1] \* i)              i += step          ax.plot(x, y, color='blue', label="Единичный вес")          res = self.get\_res\_coeffs(n, self.ro)          y = []          x = []          i = **min**(self.x)          while (i <= **max**(self.x)):              x.append(i)              y.append(res[0] + res[1] \* i)              i += step          ax.plot(x, y, color='green', label="Введенные значения")          ax.set\_xlabel("Y")          ax.set\_ylabel("X")          ax.legend(loc='best')          canvas = FigureCanvasTkAgg(self.f, self)          canvas.get\_tk\_widget().grid(row=0, column=4, columnspan=20, rowspan=20)      def **get\_res\_coeffs**(self, n, ro):          m1 = np.zeros((n + 1, n + 1))          m2 = np.zeros(n + 1)          for i in **range**(n + 1):              for j in **range**(n + 1):                  m1[i][j] = self.scalar\_x\_x(i, j, ro)              m2[i] = self.scalar\_y\_x(i, ro)            res = np.linalg.solve(m1, m2)          return res      def **get\_polynoms\_change\_pow**(self):          if (self.drawed):              self.f.clear()          else:              self.drawed = True          self.f = Figure(figsize=(10, 5), dpi=93)          ax = self.f.add\_subplot(111)          ax.plot(self.x, self.y, 'x', color="r", label="Исходные точки")            n = 1            res = self.get\_res\_coeffs(n, self.ro)          y = []          x = []          step = (**max**(self.x) - **min**(self.y)) / 100          i = **min**(self.x)          while (i <= **max**(self.x)):              x.append(i)              y\_local = 0              for j in **range**(**len**(res)):                  y\_local += res[j] \* i \*\* j              y.append(y\_local)              i += step          ax.plot(x, y, color='blue', label="n=1")          n += 1          if (n <= self.N):              res = self.get\_res\_coeffs(n, self.ro)              y = []              x = []              i = **min**(self.x)              while (i <= **max**(self.x)):                  x.append(i)                  y\_local = 0                  for j in **range**(**len**(res)):                      y\_local += res[j] \* i \*\* j                  y.append(y\_local)                  i += step              ax.plot(x, y, color='m', label="n=2")          n += 1          if (n <= self.N):              res = self.get\_res\_coeffs(n, self.ro)              y = []              x = []              i = **min**(self.x)              while (i <= **max**(self.x)):                  x.append(i)                  y\_local = 0                  for j in **range**(**len**(res)):                      y\_local += res[j] \* i \*\* j                  y.append(y\_local)                  i += step              ax.plot(x, y, color='y', label="n=3")          n += 1          if (n <= self.N):              res = self.get\_res\_coeffs(n, self.ro)              y = []              x = []              i = **min**(self.x)              while (i <= **max**(self.x)):                  x.append(i)                  y\_local = 0                  for j in **range**(**len**(res)):                      y\_local += res[j] \* i \*\* j                  y.append(y\_local)                  i += step              ax.plot(x, y, color='green', label="n=4")          n += 1          if (n <= self.N):              res = self.get\_res\_coeffs(n, self.ro)              y = []              x = []              i = **min**(self.x)              while (i <= **max**(self.x)):                  x.append(i)                  y\_local = 0                  for j in **range**(**len**(res)):                      y\_local += res[j] \* i \*\* j                  y.append(y\_local)                  i += step              ax.plot(x, y, color='purple', label="n=5")          n += 1          if (n <= self.N):              res = self.get\_res\_coeffs(n, self.ro)              y = []              x = []              i = **min**(self.x)              while (i <= **max**(self.x)):                  x.append(i)                  y\_local = 0                  for j in **range**(**len**(res)):                      y\_local += res[j] \* i \*\* j                  y.append(y\_local)                  i += step              ax.plot(x, y, color='pink', label="n=6")          ax.set\_xlabel("Y")          ax.set\_ylabel("X")          ax.legend(loc='best')          canvas = FigureCanvasTkAgg(self.f, self)          canvas.get\_tk\_widget().grid(row=0, column=4, columnspan=20, rowspan=20)        def **scalar\_x\_x**(self, pow\_1, pow\_2, ro):          res = 0.0          for i in **range**(self.N):              res += ro[i] \* self.x[i] \*\* (pow\_1 + pow\_2)          return res      def **scalar\_y\_x**(self, pow\_1, ro):          res = 0.0          for i in **range**(self.N):              res += ro[i] \* self.y[i] \* self.x[i] \*\* pow\_1          return res      def **init\_ui**(self):          self.drawed = False          amount\_label = Label(self, text="Количество точек функции:")          amount\_label.grid(row=0, column=0, columnspan=2)          self.amount\_entry = Entry(self)          self.amount\_entry.grid(row=0, column=2)          amount\_button = Button(self, text="Ввести", command=self.enter\_amount)          amount\_button.grid(row=1, column=0)          self.mode = ttk.Combobox(self, values = [              "Изменение степени полинома",              "Изменение веса точек"])          self.mode.current(0)          self.mode.configure(width=50)          self.mode.grid(column=0, row=2, columnspan=4, sticky=N)  if \_\_name\_\_ == '\_\_main\_\_':      Window = Tk()      ex = SquareApproximation(Window)      ex.grid(row=0, column=0)      Window.geometry("1900x800")      Window.mainloop() |

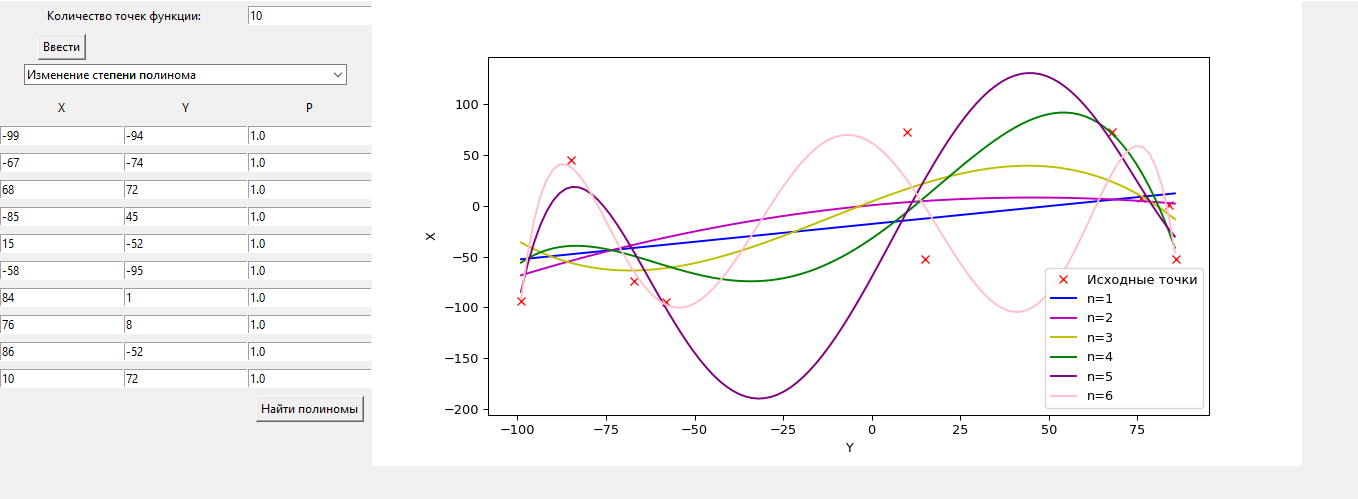
**Результаты работы.**

Графики, построенные по аналогии с рис.1 в тексте Лекции №4: точки - заданная табличная функция, кривые - найденные полиномы.

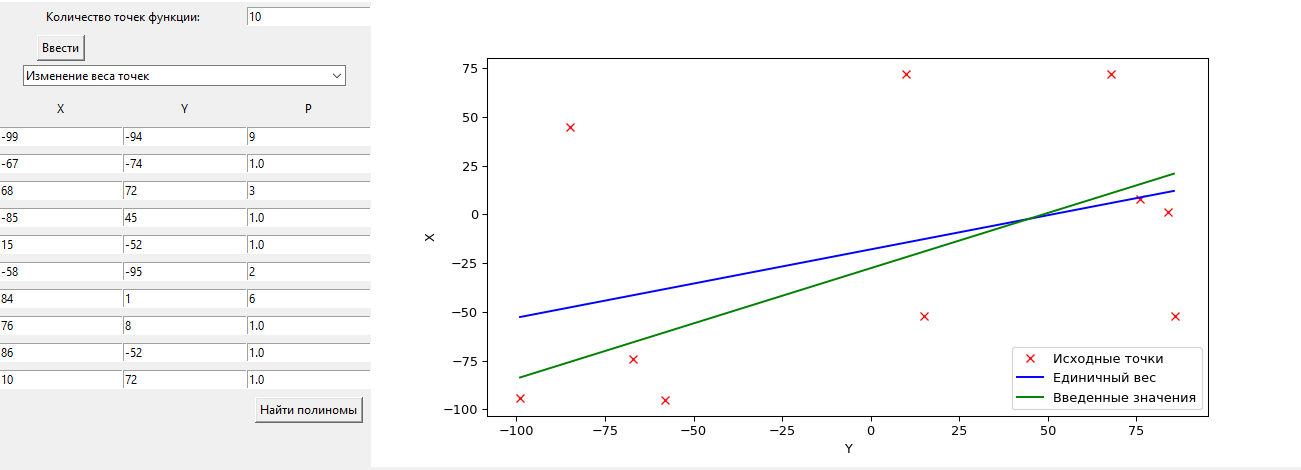
Обязательно приводить таблицы, по которым работала программа.

При каких исходных условиях надо представить результаты в отчете?

1. Веса всех точек одинаковы и равны, например, единице. Обязательно построить полиномы степеней n=1 и 2. Можно привести результаты и при других степенях полинома, однако, не загромождая сильно при этом рисунок.



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 ρi =1 для всех узлов, а другая- назначенным разным весам точек. Информацию о том, какие именно веса были использованы в расчете обязательно указать, чтобы можно было проконтролировать работу программы (лучше это сделать в виде таблицы).



**Вопросы при защите лабораторной работы.**

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

При задании степени полинома, на единицу меньшего количества узлов таблицы, график квадратичного приближения пройдет по всем исходным точкам, однако между ними будут наблюдаться большие скачки значений.

1. Будет ли работать ваша программа при n ≥ N ? Что именно в алгоритме требует отдельного анализа данного случая и может привести к аварийной остановке?

Программа будет работать, однако в процессе повышения степени может получиться СЛАУ, определитель которой равен нулю. В таком случае, вычислить коэффициенты не удастся, и программа аварийно завершится.

1. Получить формулу для коэффициента полинома a0 при степени полинома n=0. Какой смысл имеет величина, которую представляет данный коэффициент?

Коэффициент a0 будет вычисляться по формуле , т.е. a0 равен сумме весов точек. Обратная величина к нему – относительная погрешность заданной функции.

1. Записать и вычислить определитель матрицы СЛАУ для нахождения коэффициентов полинома для случая, когда n=N=2. Принять все ρi =1.

Сразу вычислим и подставим значения по формулам

1. Построить СЛАУ при выборочном задании степеней аргумента полинома , причем степени n и m в этой формуле известны.

Каждая строка СЛАУ для такого полинома будет иметь вид, причем i принимает значения от 0 до max(m, n).