|  |  |
| --- | --- |
| **Gerb-BMSTU_01** | **Министерство науки и высшего образования Российской Федерации**  Калужский филиал  федерального государственного бюджетного  образовательного учреждения высшего образования  ***«Московский государственный технический университет имени Н.Э. Баумана (национальный исследовательский университет)»***  ***(КФ МГТУ им. Н.Э. Баумана)*** |

**ФАКУЛЬТЕТ** \_***ИУК «Информатика и управление»***

**КАФЕДРА** \_\_***ИУК4 «Программное обеспечение ЭВМ, информационные технологии»***

**ЛАБОРАТОРНАЯ РАБОТА**

**«Приближение функций»**

**ДИСЦИПЛИНА: «Методы обработки информации»**

|  |  |  |
| --- | --- | --- |
| Выполнил: студент гр. ИУК4-42Б | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_Петроченков И. А.\_\_)  (Подпись) (Ф.И.О.) |
| Проверил: | | \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_ (\_\_\_Никитенко У.В.\_\_\_)  (Подпись) (Ф.И.О.) |
| Дата сдачи (защиты):  Результаты сдачи (защиты): | | |
|  | - Балльная оценка:  - Оценка: | |
| Калуга, 2024 г. | | |

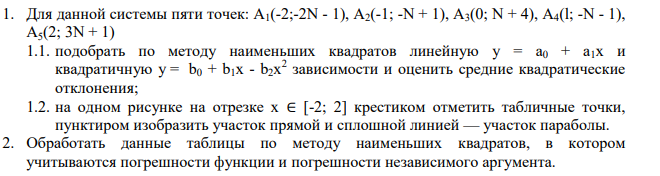
**Цели**

Применить на практике полученные знания о приближении функций методами наименьших квадратов и МНК, учитывающего погрешности зависимой и независимой переменных.

**Задачи:**

1. Выполнить линейную и квадратичную аппроксимации функций, заданных таблицами значений;
2. Выполнить аппроксимацию методом МНК, учитывающим погрешности зависимой и независимой переменных.

**Задание**

****

**Листинг кода**

from math import sqrt

import matplotlib.pyplot as plt

import numpy as np

def linear\_approx(x: list, y: list):

coeffs = np.transpose(np.array([[c for c in x],[1 for \_ in range(len(x))]]));

results = np.transpose(np.array([v for v in y]));

coeffs\_T = np.transpose(coeffs);

normal\_coeffs = coeffs\_T.dot(coeffs);

normal\_result = coeffs\_T.dot(results);

return np.linalg.solve(normal\_coeffs, normal\_result);

def quadratic\_approx(x: list, y: list):

m = 2;

results = [0 for \_ in range(m+1)];

for i in range(m, -1, -1):

for j in range(len(x)):

results[m-i] += y[j] \* pow(x[j], i);

coeffs = [[0 for \_ in range(m+1)] for \_ in range(m+1)]

for i in range(m, -1, -1):

for j in range(m, -1, -1):

for k in range(len(x)):

coeffs[m-i][m-j] += pow(x[k], i+j);

coeffs\_arr = np.array(coeffs)

results\_arr = np.array(results)

return np.linalg.solve(coeffs\_arr, np.transpose(results\_arr));

def mnk\_deviation(x: list, y: list):

P = 0;

Sx = 0

Sy = 0

div = 0;

for i in range(len(x)):

Sx += x[i];

Sy += y[i];

P += x[i]\*\*2;

P -= y[i]\*\*2;

div += x[i]\*y[i];

x2 = (Sx\*\*2)/len(x);

y2 = (Sy\*\*2)/len(x);

P -= x2;

P += y2;

div -= Sx\*Sy/len(x);

P /= div;

B1 = [(-P - sqrt(P\*\*2 + 4))/2,

(-P + sqrt(P\*\*2 + 4))/2]

B0 = [0, 0];

for c in range(len(B1)):

t = 0;

for i in range(len(y)):

B0[c] += y[i];

for i in range(len(x)):

t += x[i];

t \*= B1[c];

B0[c] -= t;

B0[c] /= len(x);

sigma\_K = [0 for \_ in range(len(B0))];

tmp = 0;

div = 0;

for i in range(len(B0)):

for n in range(len(x)):

tmp += B1[i] \* x[n];

tmp -= y[n];

tmp += B0[i]\*\*2;

div += B1[i]\*\*2 + 1;

tmp /= div;

sigma\_K[i] += tmp;

tmp = 0;

div = 0;

sigma\_K[i] /= len(x);

sigma\_K[i] \*\*= (1/2);

return [B0, B1];

# def linear\_least\_squares(x, y):

# n = len(x)

# A = np.vstack([x, np.ones(n)]).T

# m, c = np.linalg.lstsq(A, y, rcond=None)[0]

# def f(x):

# return x\*m + c;

# return f

# def quadratic\_approximation(x, y):

# n = len(x)

# A = np.vstack([np.ones(n), x, x\*\*2]).T

# a, b, c = np.linalg.lstsq(A, y, rcond=None)[0]

# def f(x):

# return a + b\*x + c\*x\*\*2;

# return f

def plot\_function(func, x\_range, label, \_type = '-'):

x = np.linspace(x\_range[0], x\_range[1], 100)

y = func(x)

plt.plot(x, y, \_type, label=label)

#plt.title(title)

plt.grid(True)

def linear\_func(a: float, b: float):

def f(x):

return a\*x + b;

return f;

def quadratic\_func(a: float, b: float, c: float):

def f(x):

return a\*x\*\*2 + b\*x + c;

return f;

# x = np.array([0, 1, 2, 3, 4, 5])

# y = np.array([1, 3, 7, 9, 11, 14])

# plot\_function(linear\_least\_squares(x, y), (-10, 10))

# plot\_function(quadratic\_approximation(x, y), (-10, 10))

# print(linear\_approx([1, 2, 3], [1, 2, 2]))

# print(quadratic\_approx([1, 2, 3, 4, 5], [5, 15, 25, 45, 65]))

# print(mnk\_deviation([-2, -1, 1, 2], [-0.8, 0.4, 1.0, -0.6]))

if \_\_name\_\_ == "\_\_main\_\_":

n = 10;

x = [-2, -1, 0, 1, 2];

y = [-2\*n - 1, -n+1, n+4, -n-1, 3\*n+1];

linear\_coeffs = linear\_approx(x, y);

quadratic\_coeffs = quadratic\_approx(x, y);

print("Коэффиценты: ")

print(f"\tЛинейной аппроксимации: {linear\_coeffs}")

print(f"\tКвадратичной аппроксимации: {quadratic\_coeffs}")

plot\_function(linear\_func(\*linear\_coeffs), [x[0], x[-1]], 'Линейная аппроксимация', 'y-')

plot\_function(quadratic\_func(\*quadratic\_coeffs), [x[0], x[-1]], 'Квадратичная аппроксимация', 'b--',)

plt.plot(x, y, 'rx', label='Табличные точки')

plt.title('Линейная и квадратичная аппроксимации')

plt.legend()

plt.grid(True)

plt.show();

#================================================================

r = [8, 9, 10, 11, 30];

t = [20.3, 21.5, 22.1, 20.5, 19.8];

coeffs = mnk\_deviation(r, t);

print("Коэффиценты: ")

print(f"\tB0: {coeffs[0]}")

print(f"\tB1: {coeffs[1]}")

plt.plot(r, t, 'rx', label='Табличные точки')

plot\_function(linear\_func(coeffs[1][0], coeffs[0][0]), [r[0], r[-1]], 'Линейная аппроксимация 1', 'y-')

plot\_function(linear\_func(coeffs[1][1], coeffs[0][1]), [r[0], r[-1]], 'Линейная аппроксимация 2', 'g-')

plt.legend()

plt.grid(True)

plt.show();

# plt.figure(figsize=(10, 6))

# plt.plot(x, linear\_func(\*linear\_coeffs), 'b--', label=f'Линейная аппроксимация\n$a\_0={linear\_coeffs[0]:.2f}, a\_1={linear\_coeffs[1]:.2f}') # Линейная модель

# plt.plot(x, quadratic\_func(\*quadratic\_coeffs), 'g-', label=f'Квадратичная аппроксимация\n$b\_0={quadratic\_coeffs[0]:.2f}, b\_1={quadratic\_coeffs[1]:.2f}, b\_2={quadratic\_coeffs[2]:.2f}') # Квадратичная модель

# plt.xlabel('x')

# plt.ylabel('y')

# plt.title('Линейная и квадратичная аппроксимация методом наименьших квадратов')

# plt.legend()

# plt.grid(True)

# plt.show()

**Результаты работы**

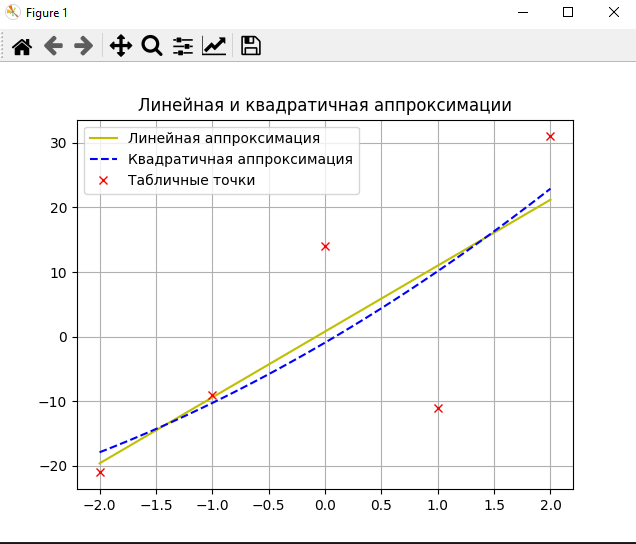
****

Рис. 1 Линейная и квадратичная аппроксимация

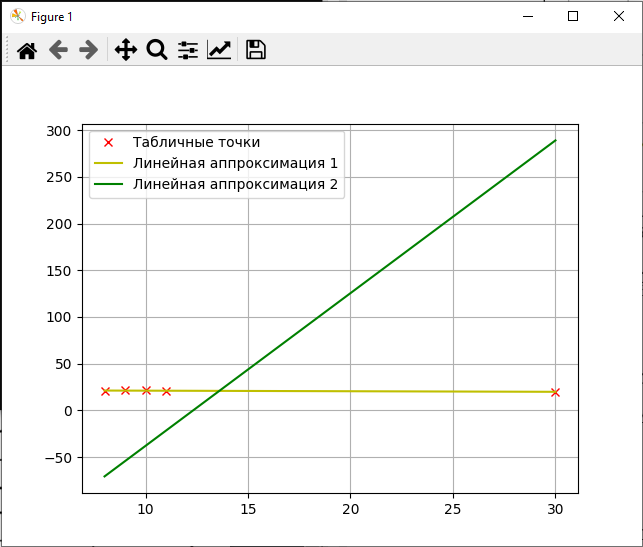


Рис. 2 Линейная аппроксимация МНК, учитывающего погрешности зависимой и независимой переменных

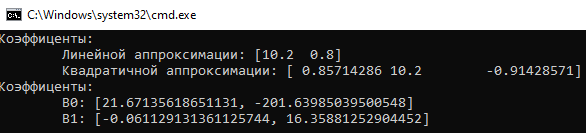


Рис. 3 Коэффициенты аппроксимаций

**Вывод**:

Применены на практике полученные знания о приближении функций методами наименьших квадратов и МНК, учитывающего погрешности зависимой и независимой переменных