**Задание 1.** Проверять, может ли полином вида

аппроксимировать функции температуры.

**Решение**

* **Алгоритм**

1. Заданы и

2. Обычным методом находим коэффициенты

Используя метод наименьших квадратов:

Данная система уравнений можно записывается в следующим матричном виде

Решением этого уравнения является:

* **Код программы** (*Python*)

# Check function: P(z) = a0 + a1.z^n1 + a2.z^n2

from math import \*

import numpy as np

import matplotlib.pyplot as plt

def f(z, a0, a1, n1, a2, n2):

return a0 + a1 \* pow(z, n1) + a2 \* pow(z, n2)

# TEST DATA

x = [ 0.05, 0.10, 0.15, 0.20, 0.25, 0.30, 0.35, 0.40, 0.45, 0.50, 0.55, 0.60, 0.65, 0.70, 0.75, 0.80, 0.85, 0.90, 0.95, 1.00]

y = [30000, 29540, 28705, 28056, 27203, 25899, 23181, 20000, 19305, 18990, 18392, 17530, 16900, 13593, 11999, 9312, 6812, 4012, 3500, 3000]

x = np.asarray(x)

y = np.asarray(y)

# Settings graphic

plt.xlabel("X")

plt.ylabel("Y")

plt.grid(True)

# Draw source data

plt.scatter(x, y)

plt.plot(x, y)

# test symlog

# plt.plot(x, y - y.mean())

# plt.yscale('symlog', linthreshy=0.01)

# Setup n1, n2

n2 = 50.0

d = 100.0

n1 = n2 - d

err = 1000000000

newY = []

\_N1 = -40.0

\_N2 = 40.0

for i in range(0, int(d \* 10) - 1):

n1 = n1 + 0.1

# Using zip to calculate sum of arrays

m\_11 = len(x)

m\_12 = np.sum(np.power(x, n1))

m\_13 = np.sum(np.power(x, n2))

m\_21 = m\_12

m\_22 = np.sum(np.power(x, 2 \* n1))

m\_23 = np.sum(np.power(x, n1 + n2))

m\_31 = m\_13

m\_32 = m\_23

m\_33 = np.sum(np.power(x, 2 \* n2))

b\_1 = np.sum(y)

b\_2 = np.sum(ty \* pow(tx, n1) for ty, tx in zip(y, x))

b\_3 = np.sum(ty \* pow(tx, n2) for ty, tx in zip(y, x))

M = np.array([[m\_11, m\_12, m\_13],

[m\_21, m\_22, m\_23],

[m\_31, m\_32, m\_33]])

B = np.array([b\_1, b\_2, b\_3])

# Solution

A = np.linalg.solve(M, B)

# Build up our figure

vect = np.vectorize(f)

new\_err = np.sum(pow(ty - tz, 2.0) for ty, tz in zip(y, vect(x, A[0], A[1], n1, A[2], n2)))

if new\_err < err:

err = new\_err

newY = vect(x, A[0], A[1], n1, A[2], n2)

\_N1 = n1

\_N2 = n2

# print("N1 = ", n1, '\tN2 = ', n2, '\terror: ', new\_err)

# Draw source data

print("Output: N1 = ", \_N1, '\tN2 = ', \_N2, '\terror: ', err)

plt.title("Modeling data")

plt.scatter(x, y)

plt.plot(x, y)

plt.scatter(x, newY)

plt.plot(x, newY)

# plt.show()

plt.savefig('03.png', dpi=720)

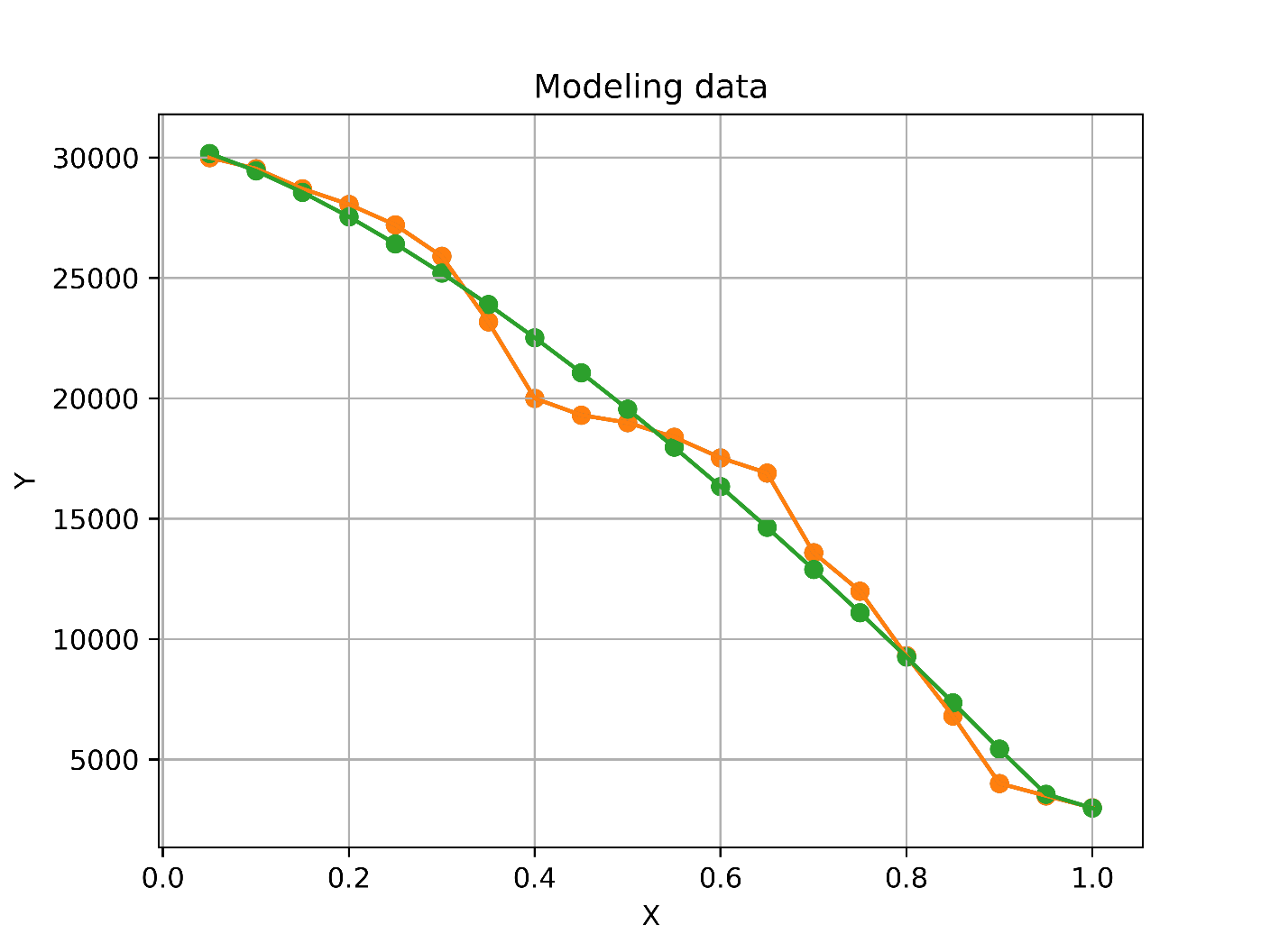
* **Эксперименты**

1. Замечание на рисуках:

* Линия оранжевая – данная;
* Линия зеленая – аппроксимированная;

1. **Вывод: полученная линия не имеет хорошее качество аппроксимации.**
2. **Экс. №1**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Вход |  |  |
| Выход |  |  |

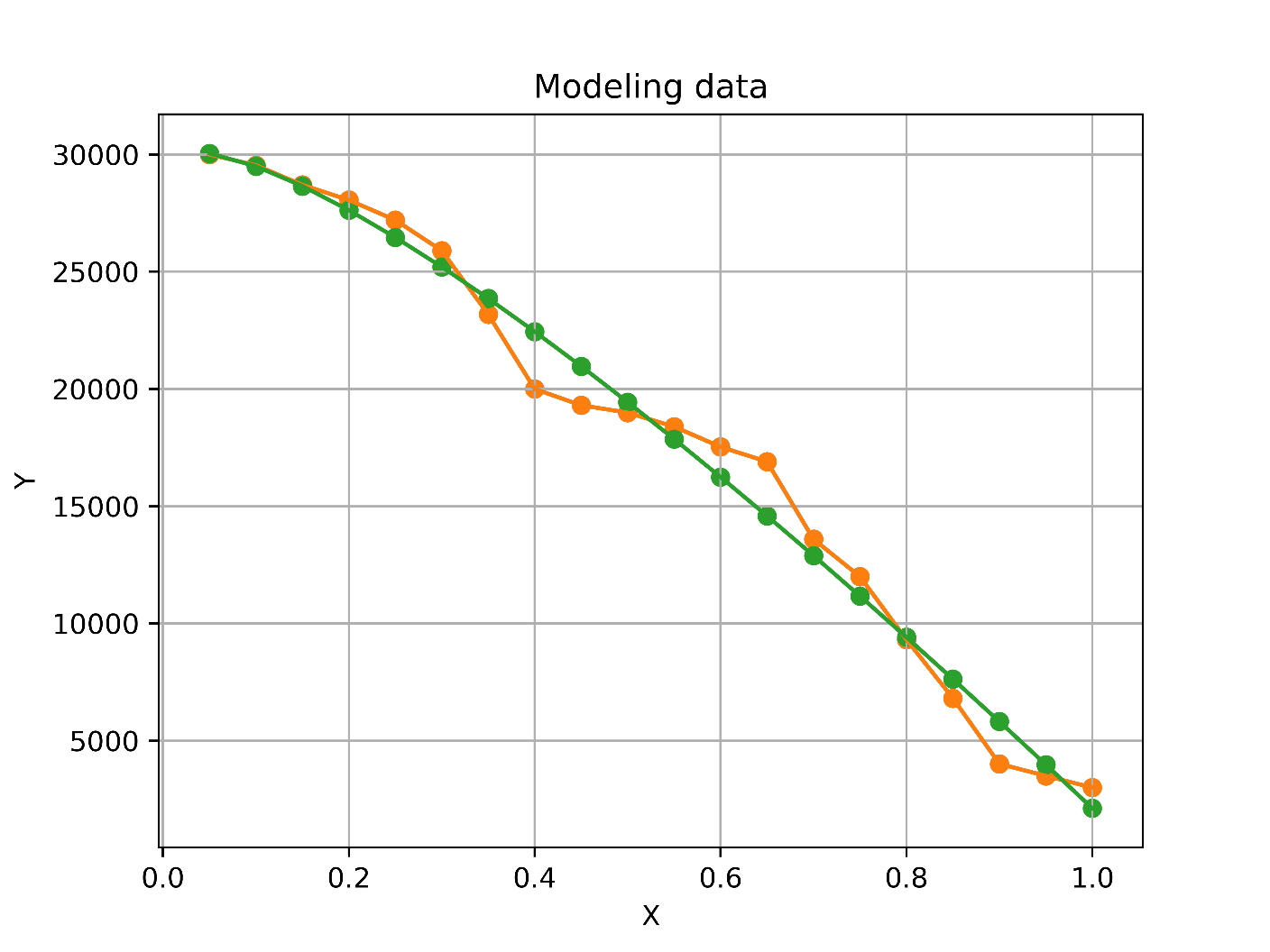


**Вывод**: качество аппроксимации не очень хорошо!

На картинке количество аппроксимировано совпадающих точек – 6 / 20.

1. **Экс. №2**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Вход |  |  |
| Выход |  |  |

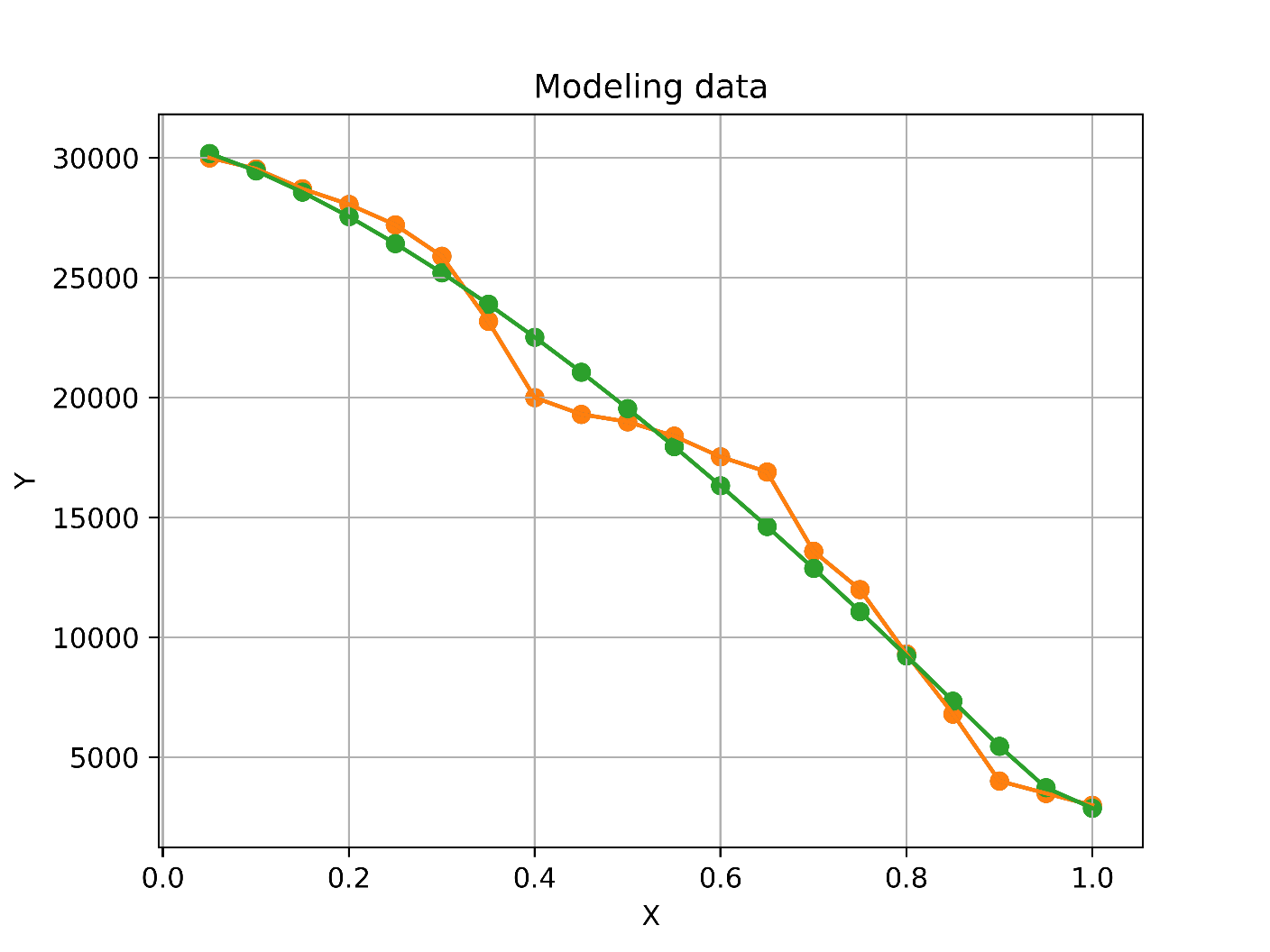


**Вывод**: качество аппроксимации не очень хорошо!

На картинке количество аппроксимировано совпадающих точек – 4 / 20.

1. **Экс. №3**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Вход |  |  |
| Выход |  |  |

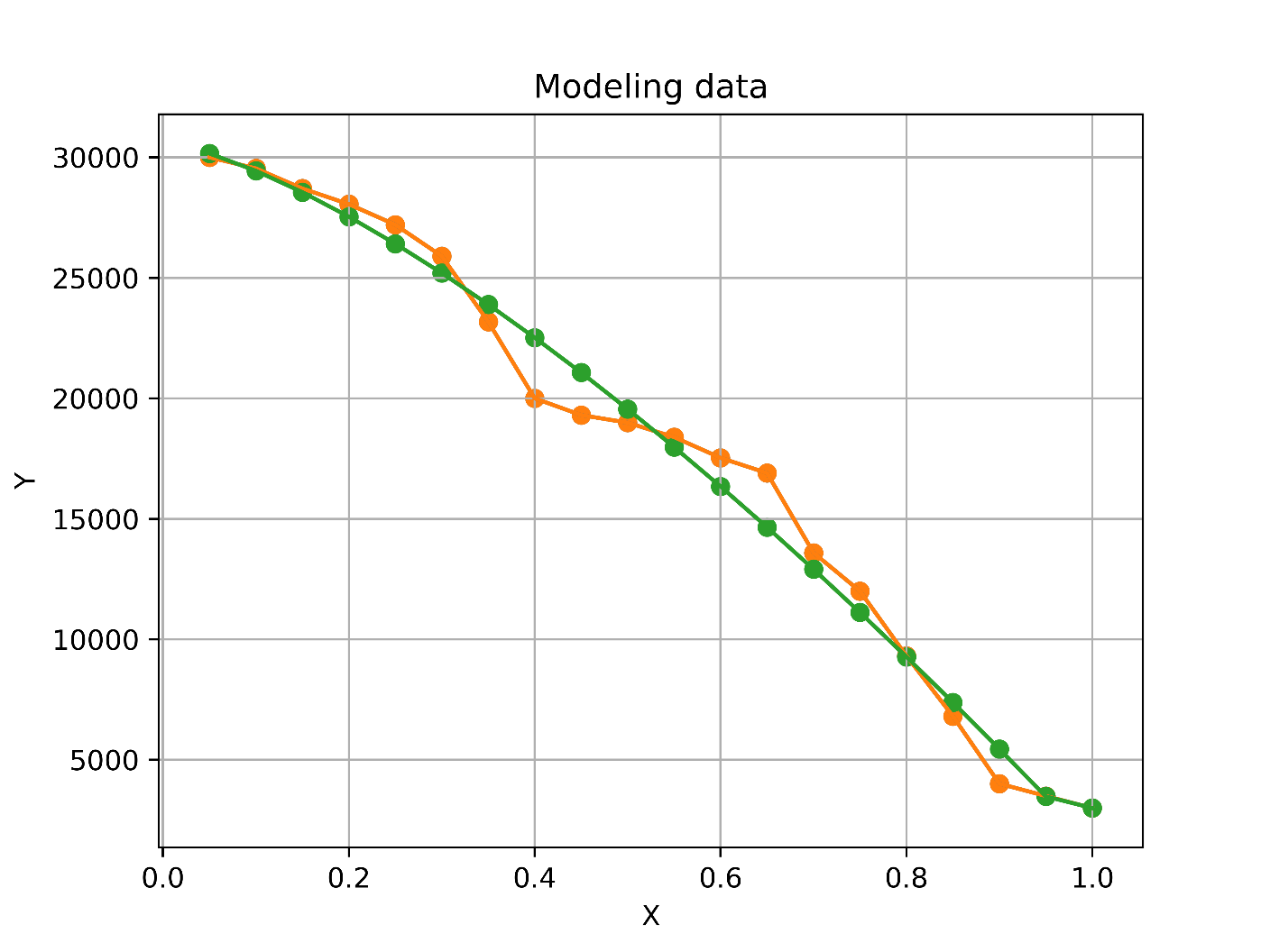


**Вывод**: качество аппроксимации не очень хорошо!

На картинке количество аппроксимировано совпадающих точек – 5 / 20.

1. **Экс. №4**

|  |  |  |
| --- | --- | --- |
|  |  |  |
| Вход |  |  |
| Выход |  |  |



**Вывод**: качество аппроксимации не очень хорошо!

На картинке количество аппроксимировано совпадающих точек – 6 / 20.