Лабораторная работа №3
Интерполирование: кубические сплайны

Выполнил студент 2 курса 3 группы ФПМИ

Сараев Владислав Максимович

Минск, 2020

Теоретические сведения

Дана функция $f(x) = (\sin(4x) - x)^3$. Необходимо произвести интерполяцию кубическими сплайнами на отрезке $[-2; 2]$ по равноотстоящим узлам с естественными граничными условиями . Интерполирование необходимо провести по N_i узлам ($N_i = 10i,\ i = 1, 2, \dots, 10$). Для каждого построение необходимо построить графики получившихся приближений и экспериментально определить максимум-норму погрешности (максимум величины $|f(x_i) - S(x_i)|, i = 1, \dots, 1000$), а также замерить затраченное время с точностью до миллисекунд и сравнить получившиеся результаты с результатами лабораторной работы №2 (по чебышевским узлам).

Для нахождения сплайна необходимо найти коэффициенты кубического многочлена $S_i = \alpha_i + \beta_i(x - x_i) + \frac{\gamma_i}{2}(x - x_i)^2 + \frac{\delta_i}{6}(x - x_i)^3, i = 1, \dots, N (1)$

Коэффициенты $\alpha_i,\ \beta_i,\ d_i$ вычисляются по следующим формулам:

$$\alpha_i = f(x_i)\ (2)$$

$$\delta_i = \frac{\gamma_i - \gamma_{i-1}}{h_i}\ (3)$$

$$\beta_i = \frac{\alpha_i - \alpha_{i-1}}{h_i} + \frac{\gamma_i}{2}h_i - \frac{\delta_i}{6}h_i^2\ (4)$$

Для нахождения коэффициентов c_i необходимо решить СЛАУ с трехдиагональной матрицей:

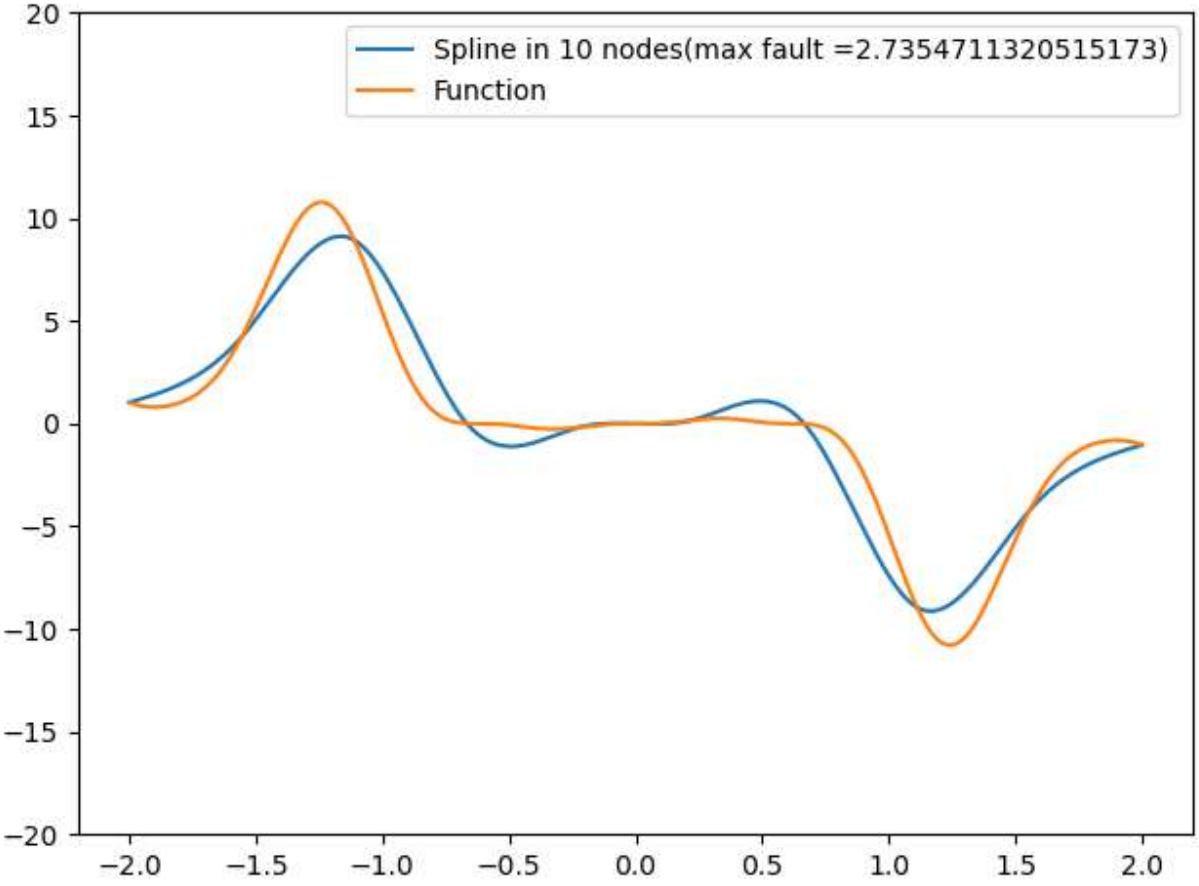
$$\gamma_{i-1}h_i + 2\gamma_i(h_i + h_{i+1}) + \gamma_{i+1}h_{i+1} = 6\left(\frac{\alpha_{i+1} - \alpha_i}{h_{i+1}} - \frac{\alpha_i - \alpha_{i-1}}{h_i}\right)$$

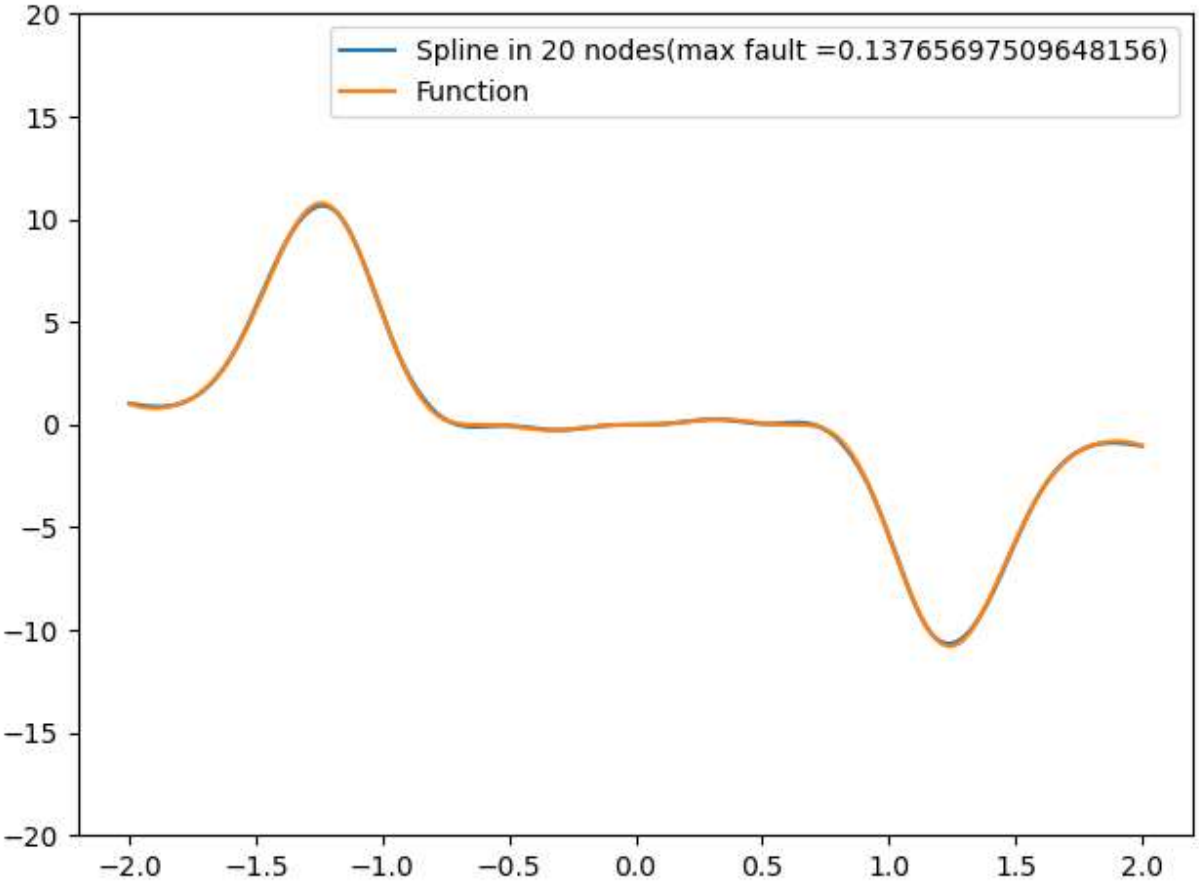
$$i = 1, \dots, N - 1\ (5)$$

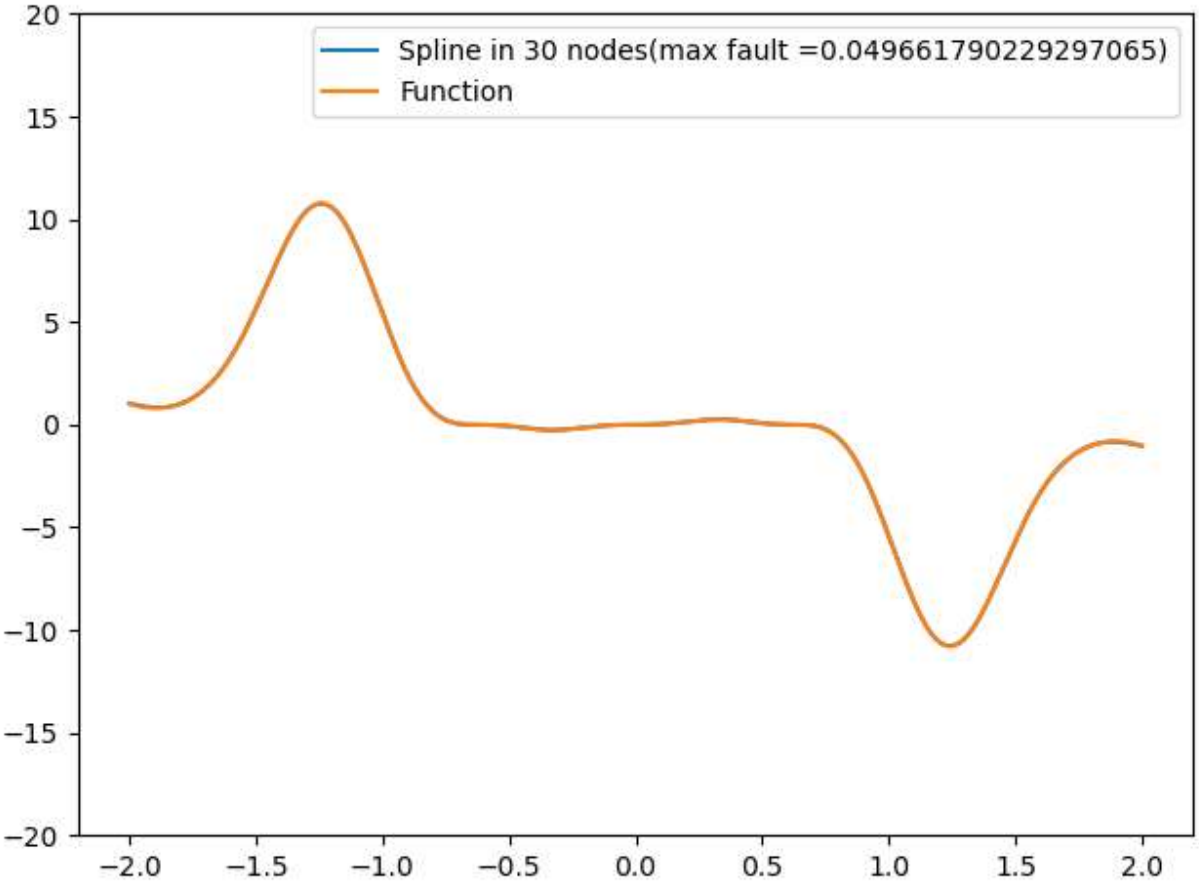
Соответственно, план выполнения задания следующий:

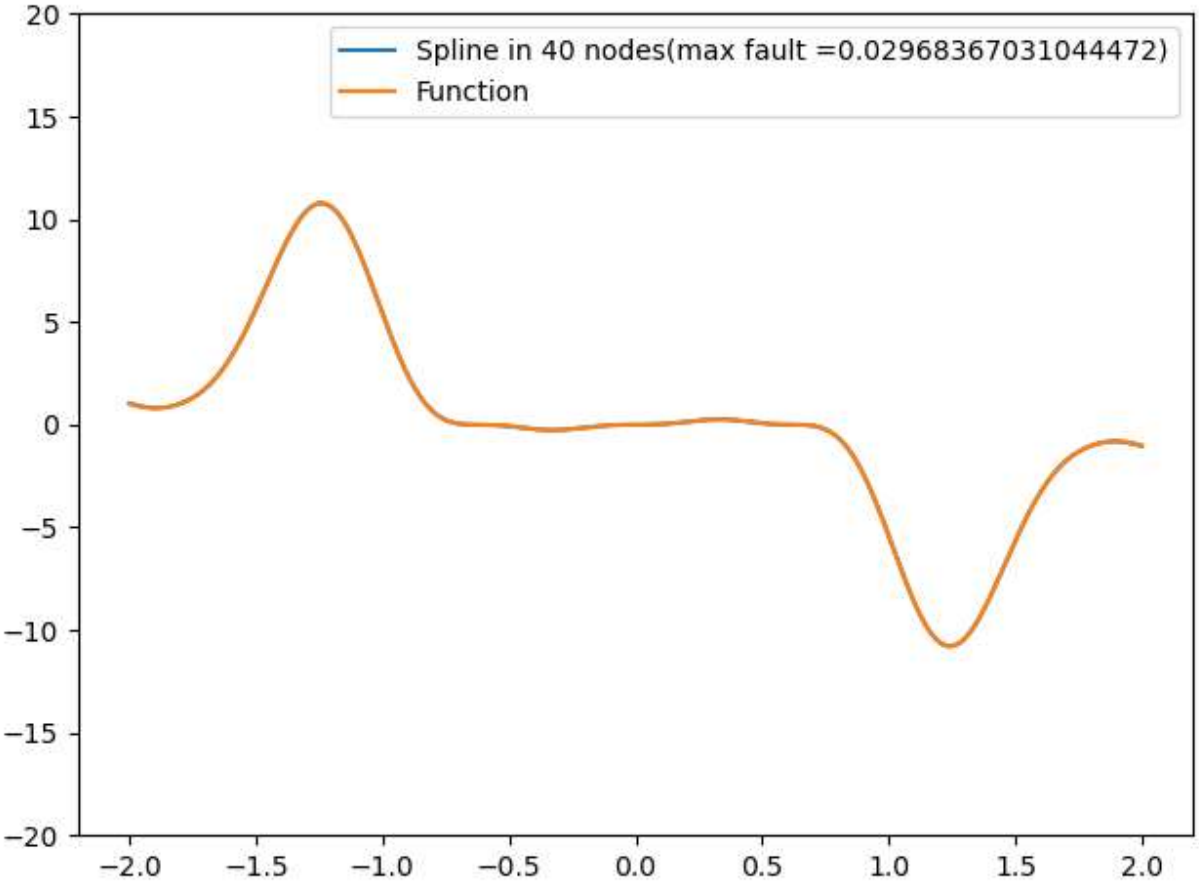
1. Разделить отрезок $[-2, 2]$ на нужное количество равноотстоящих узлов
2. Для каждого из получившихся отрезков найти коэффициенты α_i по формуле (2).
3. Решить СЛАУ (5), при условии, что $\gamma_0 = \gamma_N = 0$, например, методом прогонки, и, соответственно, найти коэффициенты γ_i.
4. Найти коэффициенты β_i, δ_i по формулам (3), (4).
5. Построить многочлен (1) по получившимся коэффициентам для каждого отрезка.
6. Найти значения получившегося сплайна в искомых точках, построить график и вычислить максимум-норму погрешности.

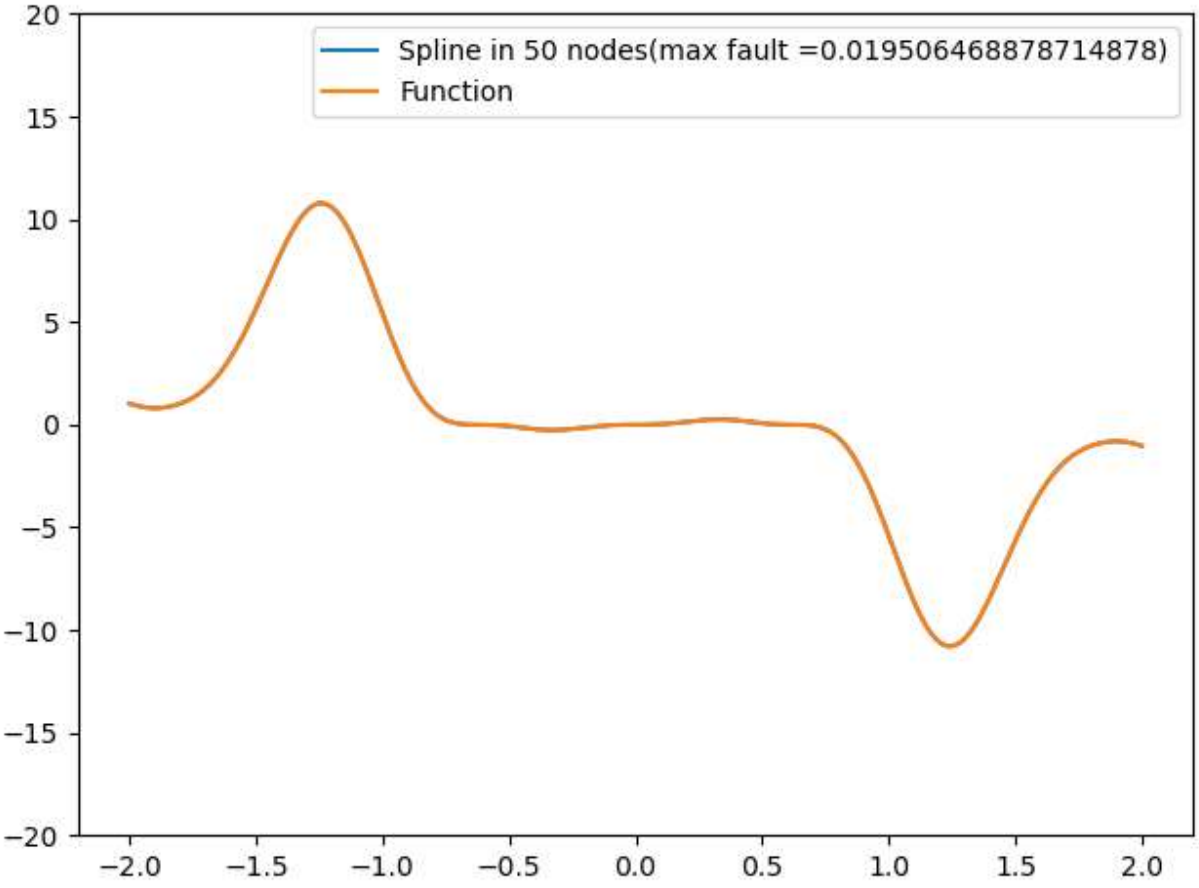
Полученное решение

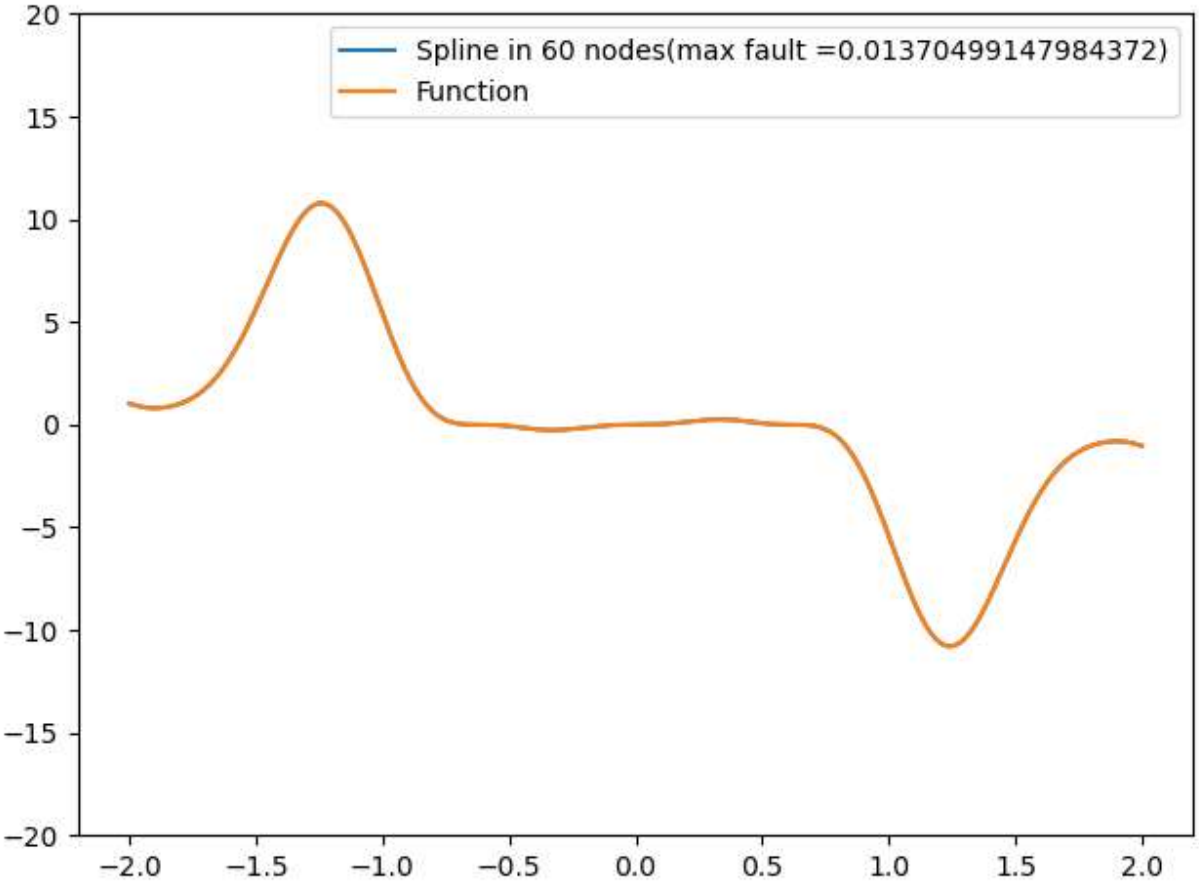


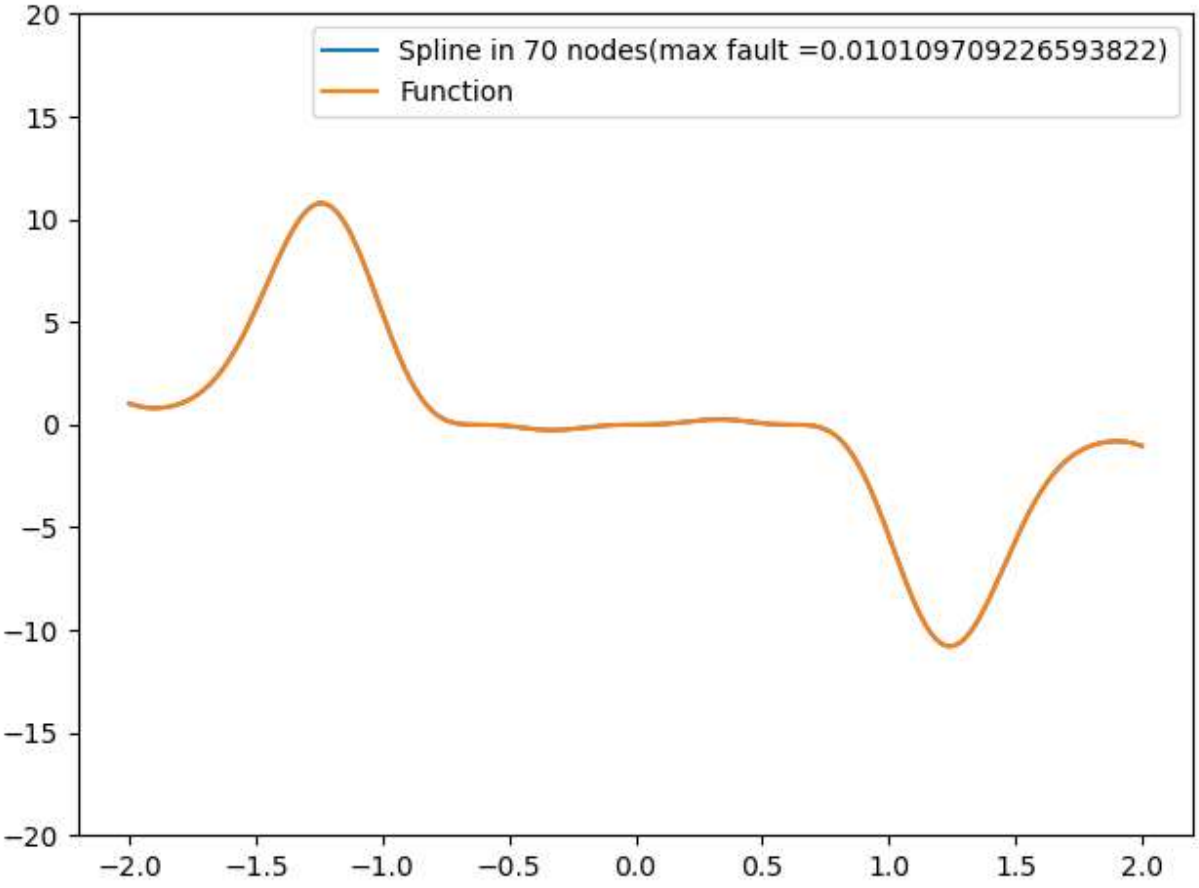


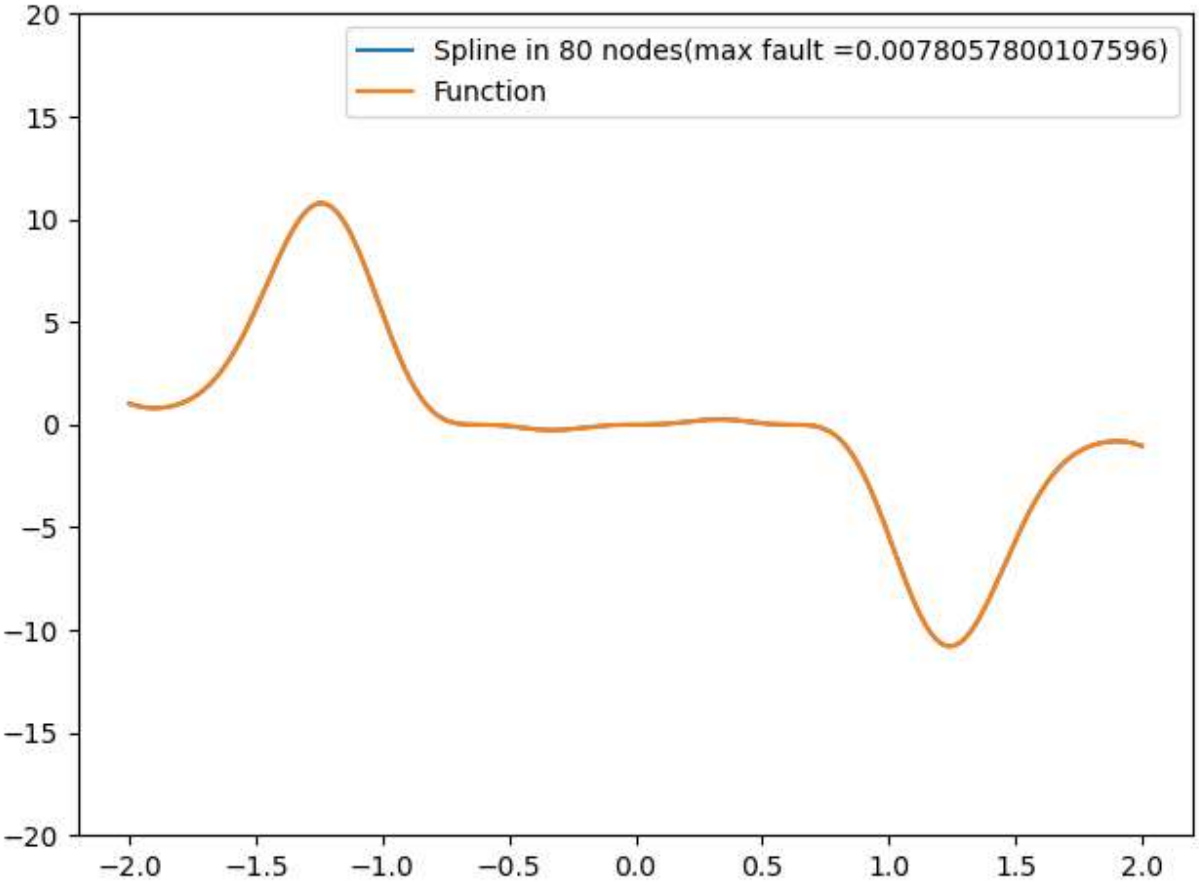

Spline in 30 nodes(max fault =0.049661790229297065)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

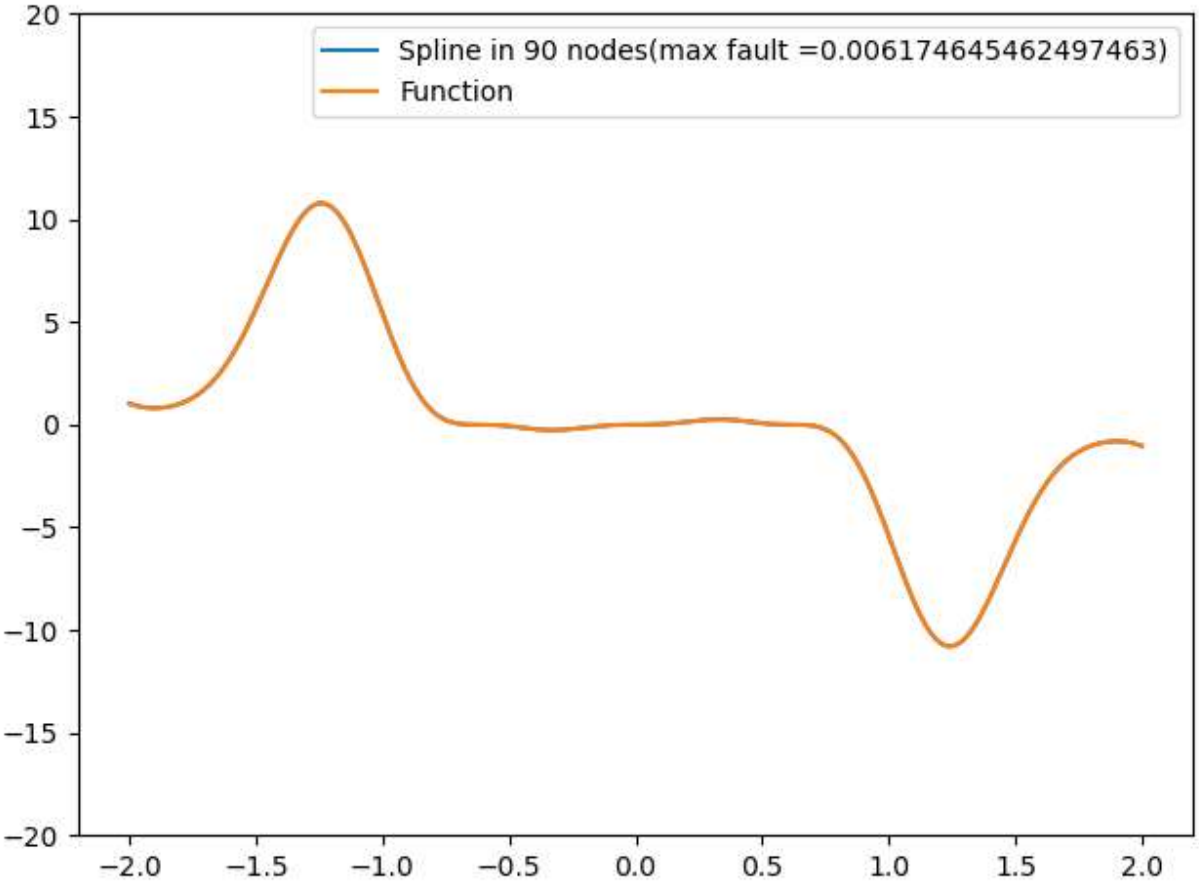

Spline in 40 nodes(max fault =0.02968367031044472)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

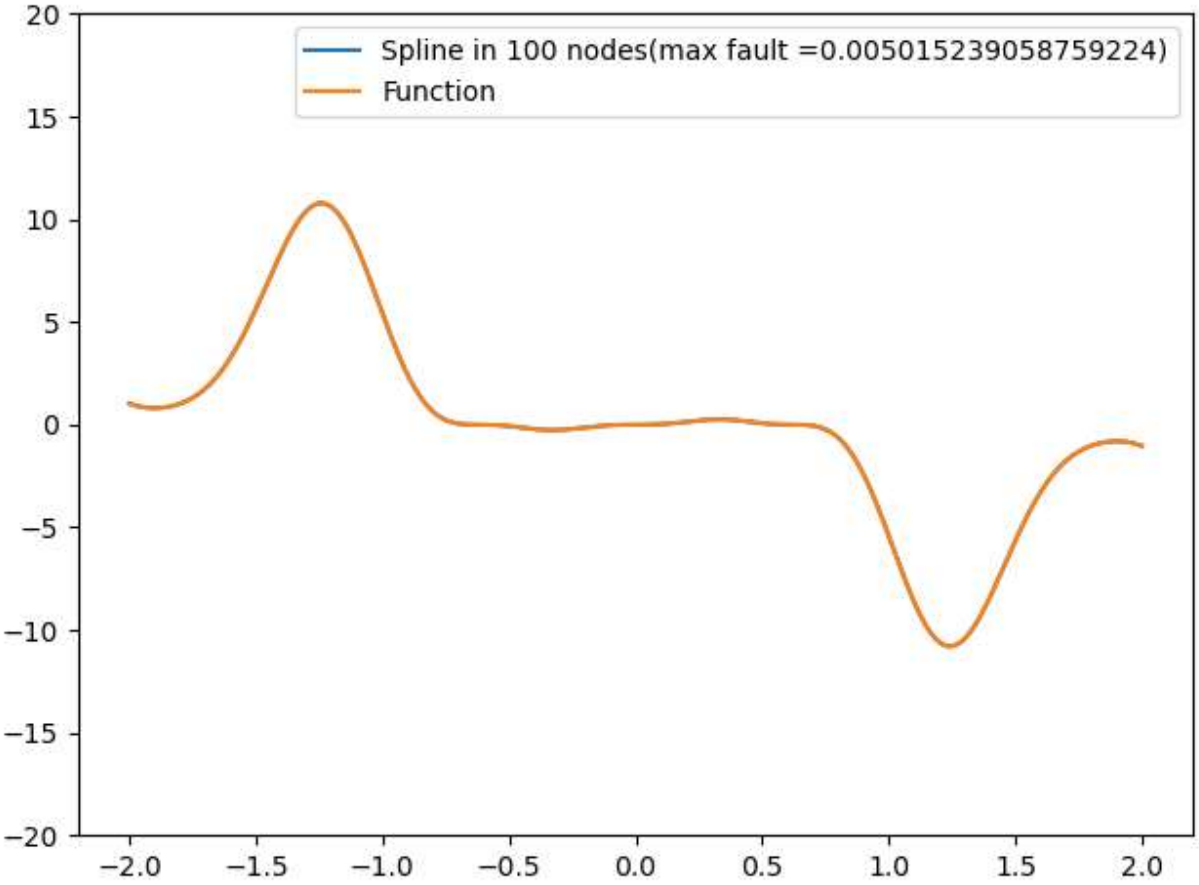

Spline in 50 nodes(max fault =0.019506468878714878)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Spline in 60 nodes(max fault =0.013704991479843 72)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Spline in 70 nodes(max fault =0.010109709226593822)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Spline in 80 nodes(max fault =0.0078057800107596)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Spline in 90 nodes(max fault =0.006174645462497463)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0


Spline in 100 nodes(max fault =0.005015239058759224)
Function
20
15
10
5
0
−5
−10
−15
−20
−2.0
−1.5
−1.0
−0.5
0.0
0.5
1.0
1.5
2.0

N	Норма (сплайн)	Норма (ЛР №2)	Время (сплайн)	Время (ЛР №2)
10	2.73547	1.24344e-14	0.005	0.154
20	0.13765	1.24344e-14	0.006	0.154
30	0.04966	1.24344e-14	0.007	0.154
40	0.02968	1.24344e-14	0.009	0.154
50	0.01950	1.24344e-14	0.008	0.154
60	0.01370	1.24344e-14	0.008	0.154
70	0.01010	1.24344e-14	0.008	0.154
80	0.00780	1.24344e-14	0.009	0.154
90	0.00617	1.24344e-14	0.008	0.154
100	0.00501	1.24344e-14	0.009	0.154

Исходный код

```
import numpy as np
from math import sin, cos, pi
import matplotlib.pyplot as plot
from time import time


def function(t):
    return (sin(4 * t) - t) ** 3


def tridiagonal_matrix_algorithm(matrix, f):
    n = len(f)
    matrix[0][2] /= matrix[0][1]
    f[0] /= matrix[0][1]
    matrix[0][1] = 1
    for i in range(1, n - 1):
        coeff = matrix[i][1] - matrix[i - 1][2] * matrix[i][0]
        matrix[i][2] /= coeff
        f[i] = (f[i] - f[i - 1] * matrix[i][0]) / coeff
        matrix[i][1] = 1
        matrix[i][0] = 0
    f[n - 1] = (f[n - 1] - f[n - 2] * matrix[n - 1][0]) / (matrix[n - 1][1] -
matrix[n - 2][2] * matrix[n - 1][0])
    matrix[n - 1][1] = 1

    for i in range(n - 2, -1, -1):
        f[i] -= f[i + 1] * matrix[i][2]
        matrix[i][2] = 0


def spline_dot(t, ai, bi, ci, di, xi):
    return ai + bi * (t - xi) + ci / 2 * ((t - xi) ** 2) + di / 6 * ((t - xi)
** 3)


def spline_dots(dots, x, a, b, c, d):
    ind = 1
    ans = []
    for i in dots:
        if i > x[ind]:
            ind += 1
        ans.append(spline_dot(i, a[ind], b[ind], c[ind], d[ind], x[ind]))
    ans = np.asarray(ans)
    return ans


L = -2
R = 2

dots = np.linspace(L, R, 1000)
count_of_dots = [10 * i for i in range(1, 11)]
faults = []
function_dots = [function(t) for t in dots]
times = []
for l in range(len(count_of_dots)):
    N = count_of_dots[l]
    x = np.linspace(L, R, N)
    y = np.empty(N)

    for i in range(0, N):
        y[i] = function(x[i])
```

```
    start = time()

    a = np.array([y[i] for i in range(0, N)])
    b = np.empty(N)
    c = np.empty(N)
    c[0] = c[N - 1] = 0
    d = np.empty(N)
    h = x[1] - x[0]

    matrix = np.array([np.array([h, 4 * h, h]) for i in range(0, N - 2)])
    matrix[0][0] = matrix[N - 3][2] = 0
    f = np.array([(a[i + 1] - 2 * a[i] + a[i - 1]) * 6 / h for i in range(1,
N - 1)])

    tridiagonal_matrix_algorithm(matrix, f)

    for i in range(1, N - 1):
        c[i] = f[i - 1]

    for i in range(1, N):
        d[i] = (c[i] - c[i - 1]) / h

    for i in range(1, N):
        b[i] = (a[i] - a[i - 1]) / h + c[i] * h / 2 - d[i] * h * h / 6

    spline = spline_dots(dots, x, a, b, c, d)
    max_fault = max(abs(spline[i] - function_dots[i]) for i in
range(len(spline)))

    end = time()

    label_info = "Spline in " + str(N) + " nodes(max fault =" +
str(max_fault) + ")"
    plot.plot(dots, spline, label=label_info)
    plot.plot(dots, function_dots, label="Function")
    plot.legend()
    plot.ylim(-20, 20)
    plot.show()
    times.append(end - start)
    faults.append(max_fault)

for i in range(len(times)):
    print(str(i + 1) + " : " + str(times[i]))

for i in range(len(faults)):
    print(str(i + 1) + " : " + str(faults[i]))
```