БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

Отчёт по лабораторной работе №3 «Интерполяционный кубический сплайн»

Выполнил:
Гаргома А. О.

Преподаватель:
Горбачева Ю. Н.

Минск, 2020

Содержание

1 Постановка задачи

Вычислить значения заданной функции $f(x)$ в узлах интерполяции $x_i = a + ih, i = \overline{0, n}$ на отрезке $[a, b]$ с шагом $h = \frac{b-a}{n}$. По полученной таблице $x_,, f(x_i)$ построить интерполяционный кубический сплайн $S_3(x)$ с дополнительными условиями:

$$S_3'(a) = f'(a),\ S_3'(b) = f'(b) \tag{1}$$

$$S_3''(a) = f''(a),\ S_3''(b) = f''(b) \tag{2}$$

$$S_3''(a) = 0,\ S_3''(b) = 0 \tag{3}$$

Убедиться, что значения функций у узлах интерполяции совпадают со значениями сплайна для всех типов дополнительных условий. В точках $\overline{x_j} = a + (j + 0.5)h,\ j = \overline{0, n-1}$ вычислить значения сплайна для всех типов дополнительных условий и сравнить со значениями функции $f(x)$ в этих точках, т.е. найти

$$\max_{j=0,\ n-1} |f(\overline{x_j}) - S_3(\overline{x_j})| \tag{4}$$

В одной системе координат построить график функции $f(x)$ и графики кубического сплайна для трёх типов дополнительных условий.

Функция $f(x) = \cos(x^2 + x)$, $n = 20, [a, b] = [-2, 2]$

2 Теория

Кубический сплайн имеет вид:

$$P_{i,3}(x) = a_i + \beta_i(x - x_{i-1}) + \frac{\gamma_i(x - x_{i-1})^2}{2} + \frac{\delta_i(x - x_{i-1})^3}{6} \tag{5}$$

Коэффициенты $\alpha_i, \beta_i,\ \gamma_i,\ \delta_i$ находится из соотношений

$$\alpha_i = f(x_i) \tag{6}$$
$$\beta_i = \frac{f(x_i) - f(x_{i-1})}{h_i} + \frac{(2 * \gamma_i + \gamma_{i-1}) * h_i}{6} \tag{7}$$
$$\delta_i = \frac{\gamma_i - \gamma_{i-1}}{h_i} \tag{8}$$
$$c_i * \gamma_{i-1} + 2 * \gamma_i + e_i * \gamma_{i+1} = b_i \tag{9}$$
$$c_i = \frac{h_i}{h_i + h_{i+1}} \tag{10}$$
$$e_i = \frac{h_{i+1}}{h_i + h_{i+1}} \tag{11}$$
$$b_i = 6 * f[x_{i-1}, x_i, x_{i+1}] \tag{12}$$
$$h_i = x_i - x_{i-1}, i = \overline{1, n-1} \tag{13}$$
$$\tag{14}$$

1. $S_3'(a) = f'(a),\ S_3'(b) = f'(b)$

$$2 * \gamma_0 + \gamma_1 = 6 * f[x_0, x_1, x_2] \tag{15}$$
$$\gamma_{n-1} + 2 * \gamma_n = 6 * f[x_{n-1}, x_n, x_n] \tag{16}$$

2. $S_3''(a) = f''(a),\ S_3''(b) = f''(b)$

$$\gamma_0 = \frac{f''(a)}{2},\ \gamma_n = \frac{f''(b)}{2} \tag{17}$$

3. $S_3''(a) = 0,\ S_3''(b) = 0$

$$\gamma_0 = \gamma_n = 0 \tag{18}$$

3 Программа

```
#!/usr/bin/env python
# coding: utf-8

# In[1]:


import numpy as np
import matplotlib.pyplot as plt
import math
import sympy as sp
from tabulate import tabulate
get_ipython().run_line_magic('matplotlib', 'inline')


# In[2]:


a,b = -2,2
```

```
n = 20
h = (b-a)/n
length = abs(b-a)
def func(x):
    return math.cos(x+x**2)
x = np.linspace(a,b,10000)
y = np.array(list(map(func,x)))
t = sp.symbols('t')
sfunc = sp.cos(t+t**2)
drvtv1 = sp.lambdify(t,sp.diff(sfunc,t))
drvtv2 = sp.lambdify(t,sp.diff(sfunc,t,t))

nodes = np.array([a+i*h for i in range(n+1)])
vfunc = np.vectorize(func)
values = vfunc(nodes)
coefs = np.zeros((n+1,n+1)) # матрица коэффициентов гамма
column = np.zeros((n+1,1))
h_i = np.array([nodes[i+1]-nodes[i] for i in range(n)]) # x_i-x_{i-1}
differences = np.array([(values[i+1]-values[i])/h_i[i] for i in range(n)]) # разделенные разности
for i in range(1,n):
    coefs[i][i-1] = h_i[i-1]/(h_i[i-1]+h_i[i])
    coefs[i][i] = 2
    coefs[i][i+1] = h_i[i]/(h_i[i-1]+h_i[i])
    column[i] = 6*(differences[i]-differences[i-1])/(h_i[i-1]+h_i[i])



# In[4]:


def get_spline(gamma):
    delta = np.zeros((n))
    betta = np.zeros((n))
    alpha = values[1:]
    for i in range(n):
        delta[i] = (gamma[i+1]-gamma[i])/h_i[i]
        betta[i] = (values[i+1]-values[i])/h_i[i]+(2*gamma[i+1]+gamma[i])*h_i[i]/6
    def spline(x):
        idx = int((x-a)/h)
        if idx == n:
            idx = n-1
        return (alpha[idx]+betta[idx]*(x-nodes[idx+1])+gamma[idx+1]/2*(x-nodes[idx+1])**2+delta[idx]/6*(x-nodes[idx+1])**3)[0]

    return spline


# # 1 дополнительные условия s'(a) = f'(a), s'(b) = f'(b),

# In[5]:


coefs[0][0] = 2
coefs[0][1] = 1
column[0] = 6/h_i[0]*(differences[0]-drvtv1(a))

coefs[n][n-1] = 1
coefs[n][n] = 2
```

```
column[n] = 6/h_i[n-1]*(drvtv1(b)-differences[n-1])
gamma1 = np.linalg.solve(coefs,column)
spline1 = get_spline(gamma1)


# # 2 дополнительные условия s''(a) = f''(a), s''(b) = f''(b)

# In[6]:


coefs[0][0] = 1
coefs[0][1] = 0
column[0] = drvtv2(a)

coefs[n][n-1] = 0
coefs[n][n] = 1
column[n] = drvtv2(b)
gamma2 = np.linalg.solve(coefs,column)
spline2 = get_spline(gamma2)


# # 3 дополнительные условия s''(a) = 0, s''(b) = 0

# In[7]:


coefs[0][0] = 1
coefs[0][1] = 0
column[0] = 0

coefs[n][n-1] = 0
coefs[n][n] = 1
column[n] = 0
gamma3 = np.linalg.solve(coefs,column)
spline3 = get_spline(gamma3)


# # Построение графиков

# In[8]:


splines = [spline1,spline2,spline3]
for i,spline in enumerate(splines):
    check_nodes = []
    for node in nodes:
        check_nodes.append(func(node) - spline(node))
    error = 0
    for node in nodes[:-1]:
        error = max(error, abs(func(node+0.5*h)-spline(node+0.5*h)))
    print('Разница значений функции и сплайна в узлах интерполирования:\n ', check_nodes)
    print(f'Погрешность интерполяции сплайном в серединах отрезков: {error}')
    fig = plt.figure(figsize=(18,10))
    plt.plot(x,y,'b',linewidth = 4,label='исходная функция')
    plt.plot(nodes,vfunc(nodes),'go')
    plt.plot(x,list(map(spline,x)),'r--', linewidth = 4, alpha = 0.5,label=f'сплайн{i+1}')
    plt.grid(True)
```

```
    plt.legend()
    plt.title(f'График исходной функции и сплайна{i+1}')
    axes = fig.axes
    axes[0].set_xticks(np.arange(-2,2.1,0.1))
    plt.xlabel('x')
    plt.ylabel('y')
    plt.savefig(f'images/{i+1}')
```

4 Результаты

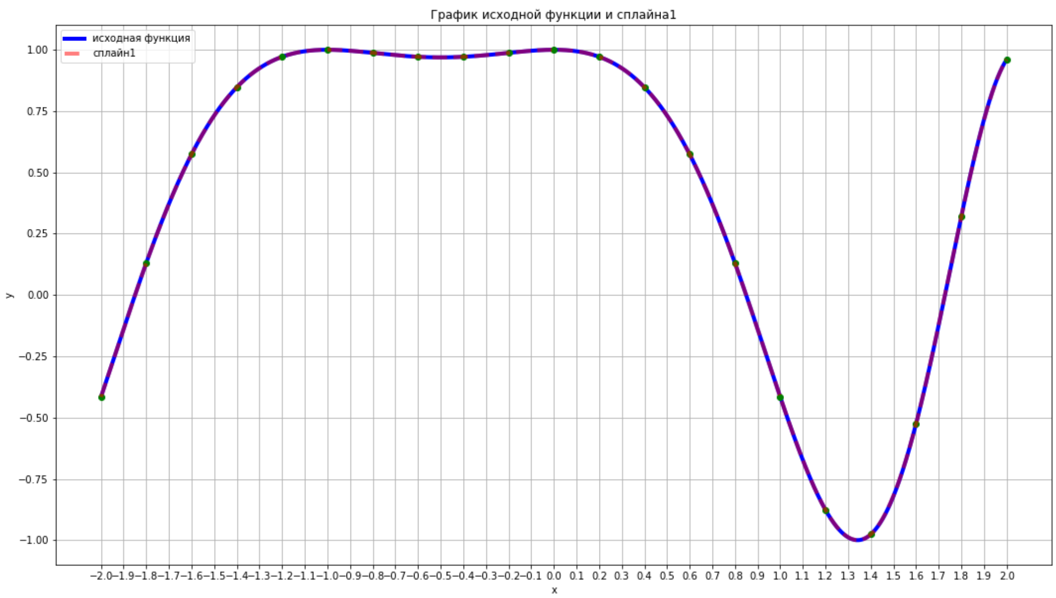

Рис. 1: Первое дополнительное условие

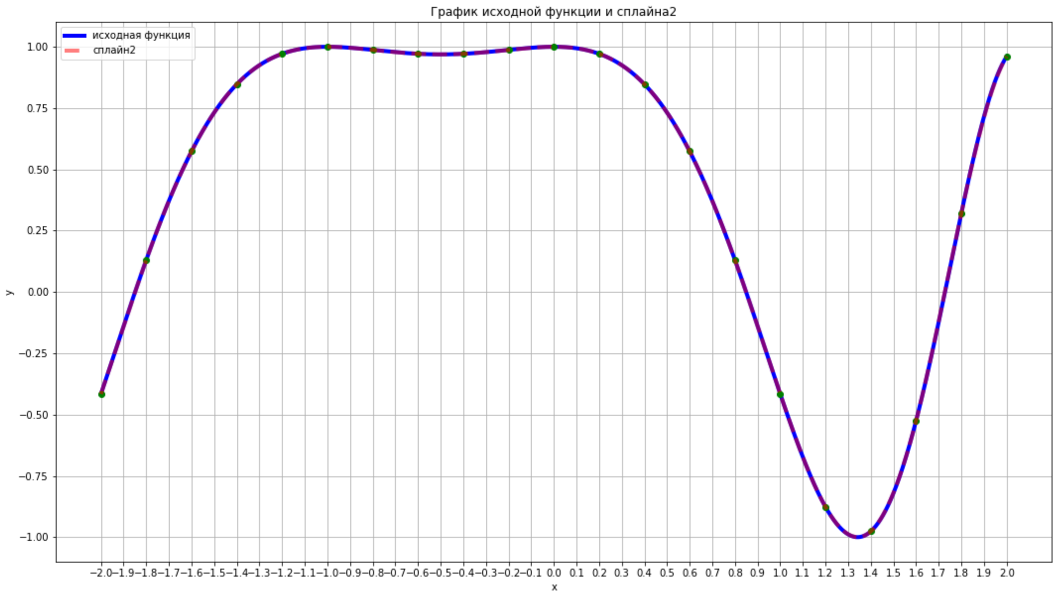

Рис. 2: Второе дополнительное условие

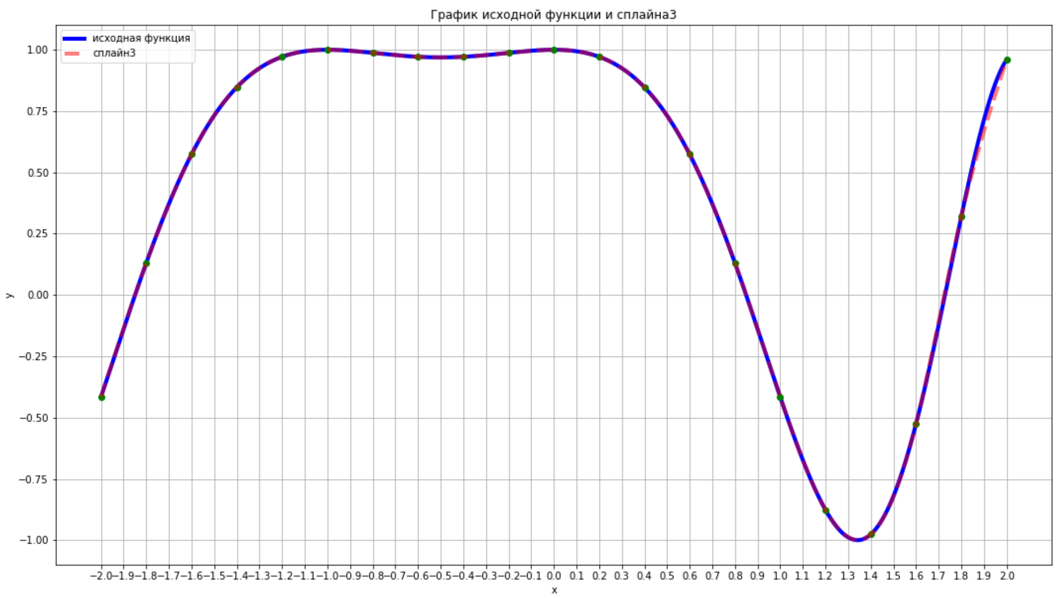

Рис. 3: Третье дополнительное условие

Сплайн	$\max_{j=0,\ n-1} \lvert f(\overline{x_j}) - S_3(\overline{x_j}) \rvert$
1	0.0019595685331641882
2	0.003589376437711289
3	0.04649748434138057

5 Выводы

Из таблицы погрешностей выше видно, что естественные граничные условия имеют большую погрешность, нежели другие граничные условия.