Министерство образования Республики Беларусь

Учреждение образования

БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ

Факультет прикладной математики и информатики

ОТЧЕТ

ПО ПРЕДМЕТУ “МЕТОДЫ ЧИСЛЕННОГО АНАЛИЗА”

НА ТЕМУ “ Интерполяционный кубический сплайн”

студентки 2 курса 2 группы

Курец Любови Олеговны

Преподаватель

Горбачева Юлия Николаевна

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 $h = (b-a)/n$. По полученной таблице $\{x_i, f(x_i)\}$ построить интерполяционный кубический сплайн $S_3(x)$ с дополнительными условиями:

1) $S_3'(a) = f'(a)$, $S_3'(b) = f'(b)$;
2) $S_3''(a) = f''(a)$, $S_3''(b) = f''(b)$;
3) $S_3''(a) = 0$, $S_3''(b) = 0$.

Убедиться, что значения функции в узлах интерполяции совпадают со значениями сплайна для всех типов дополнительных условий. В точках $\bar{x}_j = a + (j+0.5)h$, $j = \overline{0,n-1}$ вычислить значения сплайна для всех типов дополнительных условий и сравнить со значениями функции $f(x)$ в этих точках, т.е. найти $\max\limits_{j=0,n-1} \left| f(\bar{x}_j) - S_3(\bar{x}_j) \right|$. В одной системе координат построить график функции $f(x)$ и графики кубического сплайна для трех типов дополнительных условий.

№ п/п	ФИО	n	$[a,b]$	$f(x)$
3	Курец Любовь Олеговна	20	[10,100]	$\sin(\ln x)$

Краткие теоретические сведения:

Пусть на отрезке $[a,b]$ известны табличные значения функции $y = f(x)$ в точках $a = x_0 < x_1 < \ldots < x_n = b$ – узлах интерполирования.

Сплайн $L(x)$ на каждом сегменте $[x_{i-1}, x_i]$ отрезка $[a,b]$ строится в виде:

$$L_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3, \quad i = \overline{1,n}.. \qquad (1)$$

Соответственно, для всего интервала будет n кубических полиномов и коэффициентов a_i, b_i, c_i, d_i, где i - номер сплайна.

Коэффициенты определяются из условий:

1. Равенство значений сплайнов L_i и аппроксимируемой функции $y = f(x)$ в узлах :

$L_1(x_0) = y(x_0)$, $L_i(x_i) = y(x_i)$ при $i = \overline{1,n}$;

2. Гладкость совпадения сплайнов:

$$\left.\begin{aligned} L_{i-1}(x_{i-1}) &= L_i(x_{i-1}) \\ L'_{i-1}(x_{i-1}) &= L'_i(x_{i-1}) \\ L''_{i-1}(x_{i-1}) &= L''_i(x_{i-1}) \end{aligned}\right\} \text{ при } i = \overline{2,n};$$

3. Краевые условия: $L''(a) = L''(b) = 0$.

Подставляя в условия функцию сплайна $L_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$ и производные данного сплайна $L_i'(x) = b_i + 2c_i(x - x_i) + 3d_i(x - x_i)^2$ и $L_i''(x) = 2c_i + 6d_i(x - x_i)$, также, полагая $h_i = x_i - x_{i-1}$, получаем систему:

$$
\begin{cases}
a_1 - b_1 h_1 + c_1 h_1^2 - d_1 h_1^3 = y(x_0); & (2)\\
a_i = y(x_i) \qquad \text{где} \quad i = \overline{1, n}; & (3)\\
\left.\begin{aligned}
a_{i-1} &= a_i - b_i h_i + c_i h_i^2 - d_i h_i^3, \\
b_{i-1} &= b_i - 2c_i h_i + 3d_i h_i^2, \\
c_{i-1} &= c_i - 3d_i h_i,
\end{aligned}\right\} \text{ где } \quad i = \overline{2, n} & (4),\ (5),\ (6)\\
c_1 - 3d_1 h_1 = 0; & (7)\\
c_n = 0. & (8)
\end{cases}
$$

Исключая неизвестные a_i, b_i, d_i, сводим систему к решению относительно неизвестных c_i. Введем эффективный коэффициент $c_0 = 0$.

Подставляя значения (3) в равенства (2) и (4), получим выражение

$$b_i h_i - c_i h_i^2 + d_i h_i^3 = y(x_i) - y(x_{i-1}), \text{ при } i = \overline{1, n}. \tag{9}$$

Из (6) и (7), учитывая $c_0 = 0$ выразим d_i через c_i:

$$d_i = \frac{c_i - c_{i-1}}{3h_i} \text{ при } i = \overline{1, n}. \tag{10}$$

С помощью формулы (10) убираем d_i в формуле (9). Получим:

$$
\begin{cases}
d_i = \dfrac{c_i - c_{i-1}}{3h_i}, \\
b_i = \dfrac{f(x_i) - f(x_{i-1})}{h_i} + \dfrac{2c_i + c_{i-1}}{3} h_i,
\end{cases}
\quad i = \overline{1, n} \tag{11}
$$

После подстановки данных формул в выражение (5), получим выражение, с неизвестными коэффициентами c_i:

$$h_{i-1} c_{i-2} + 2(h_{i-1} + h_i) c_{i-1} + h_i c_i = 3\left[\frac{f(x_i) - f(x_{i-1})}{h_i} - \frac{f(x_{i-1}) - f(x_{i-2})}{h_{i-1}}\right]$$

$$i = \overline{2, n},\ c_0 = 0,\ c_n = 0. \tag{12}$$

Получаем замкнутую систему. Представим формулу (12) в виде

$$h_{i-1} c_{i-2} + s_i c_{i-1} + h_i c_i = r_i, \tag{13}$$

где коэффициенты s_i и r_i: $\qquad s_i = 2(h_{i-1} + h_i)$, (14)

$$r_i = 3\left[\frac{f(x_i) - f(x_{i-1})}{h_i} - \frac{f(x_{i-1}) - f(x_{i-2})}{h_{i-1}}\right], \text{ при } i = \overline{2, n}. \tag{15}$$

Пусть $c_{i-1} = k_{i-1} + l_{i-1} c_i$. (16)

Тогда $c_{i-2} = k_{i-2} + l_{i-2} c_{i-1}$. Подставим в (13). Получаем

$$c_{i-1} = \frac{r_{i-1} - h_{i-1} k_{i-2}}{s_i + h_{i-1} l_{i-2}} - \frac{h_i}{s_i + h_{i-1} l_{i-2}} c_i.$$

Откуда $l_{i-1} = -\dfrac{h_i}{s_i + h_{i-1} l_{i-2}}$, $k_{i-1} = \dfrac{r_i - h_{i-1} k_{i-2}}{s_i + h_{i-1} l_{i-2}}$, $i = \overline{2, n}$.

Учитывая, что $c_0 = 0$, получим, что $k_0 = 0$, $l_0 = 0$.

В результате получим:

1. Вычисляем $s_i = 2(h_{i-1} + h_i)$, (14)

2. $r_i = 3\left[\frac{f(x_i) - f(x_{i-1})}{h_i} - \frac{f(x_{i-1}) - f(x_{i-2})}{h_{i-1}}\right]$, (15)

при $i = \overline{2, n}$.

3. Полагаем $k_0 = 0$, $l_0 = 0$. В процессе прямого хода прогонки вычисляем прогоночные коэффициенты:

$$l_{i-1} = -\frac{h_i}{s_i + h_{i-1} l_{i-2}}, \quad i = \overline{2, n} \tag{17}$$

$$k_{i-1} = \frac{r_i - h_{i-1} k_{i-2}}{s_i + h_{i-1} l_{i-2}}, \quad i = \overline{2, n}. \tag{18}$$

4. На обратном ходе имеем $c_{i-1} = k_{i-1} + l_{i-1} c_i$ из (16), $i = n, .., 2$ и учитываем, что $c_n = 0$.

5. Затем по формуле (11) вычисляем коэффициенты:

$$\begin{cases} d_i = \dfrac{c_i - c_{i-1}}{3h_i}, \\ b_i = \dfrac{f(x_i) - f(x_{i-1})}{h_i} + \dfrac{2c_i + c_{i-1}}{3} h_i, \end{cases} \quad i = 1,..n \tag{19}$$

Листинг программы:

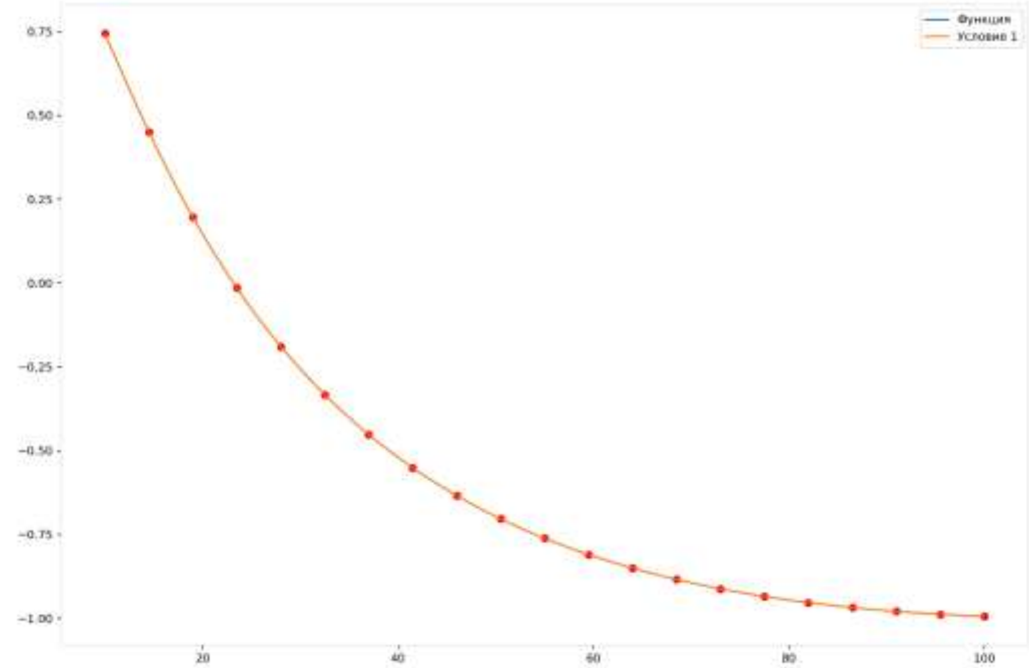
```
import numpy as np
import matplotlib.pyplot as plt

pic_count = 1
plt.rcParams['axes.linewidth'] = 0.1


def save_plt(plt):
    global pic_count
    plt.savefig("images/" + str(pic_count) + ".pdf")
    pic_count += 1
    plt.close()

#входные данные
def func(x):
    return np.sin(np.log(x))

vfunc = np.vectorize(func)
n = 20
a_, b_ = 10, 100
ddF_a = -(np.sin(np.log(10))+np.cos(np.log(10)))/100
ddF_b = -(np.sin(np.log(100))+np.cos(np.log(100)))/10000
dF_a = (np.cos(np.log(10)))/10
dF_b = (np.cos(np.log(100)))/100

def slau(a, c, b, f):

    f_copy = f.copy()

    n = len(c)
    x = [0 for i in range(n)]

    for i in range(1, n):
```

```python
        m = a[i - 1] / c[i - 1]
        c[i] -= m * b[i - 1]
        f_copy[i] -= m * f_copy[i - 1]

    x[n - 1] = f_copy[n - 1] / c[n - 1]

    for i in reversed(range(n - 1)):
        x[i] = (f_copy[i] - b[i] * x[i + 1]) / c[i]
    return x

def spline_(start, end, n, function, kind=3):
    x = np.linspace(start, end, n+1)
    y = function(x)
    a = y[:-1]
    h = (end - start) / n


    F = 3*(y[:-2] - 2*y[1:-1] + y[2:])/h**2
    dnDiag = np.ones(n)
    upGiag = np.ones(n)
    diag = 4*np.ones(n+1)

    if kind == 1:
        diag[0], diag[-1] = 2, 2
        upGiag[0], dnDiag[-1] = 1, 1
        f_1 = 3*(y[1]-y[0])/h**2 - 3*dF_a/h
        f_np1 = 3*dF_b/h - 3*(y[-1] - y[-2])/h**2
    elif kind == 2:
        diag[0], diag[-1] = 2, 2
        upGiag[0], dnDiag[-1] = 0, 0
        f_1, f_np1 = ddF_a, ddF_b
    elif kind == 3:
        diag[0], diag[-1] = 1, 1
        upGiag[0], dnDiag[-1] = 0, 0
        f_1, f_np1 = 0, 0
    else:
        raise Exception("there are only 3 kinds of splines")

    F = np.concatenate([[f_1], F, [f_np1]])

    # решаем СЛАУ методом прогонки
    C = np.array(slau(dnDiag, diag, upGiag, F))
    b = (y[1:] - y[:-1])/h - (2*C[:-1] + C[1:])*h/3
    d = (C[1:] - C[:-1])/3/h

    def spline(p):
        i = int((p - start)/(end-start)*n)
        i = min(n-1, max(0, i))
        return a[i] + b[i]*(p-x[i]) + C[i]*(p-x[i])**2 + d[i]*(p-x[i])**3
    _spline = np.vectorize(spline)

    return _spline


def s3(kind):
    plt.figure(num=None, figsize=(15, 10), dpi=180, facecolor='green',
        edgecolor='blue')

    x = np.linspace(a_, b_, 100)
    y = vfunc(x)
    xInterpNodes = np.linspace(a_, b_, n+1)
    yInterpNodes = vfunc(xInterpNodes)
    spline = spline_(a_, b_, n, vfunc, kind)
    y_spline_points = spline(x)
```

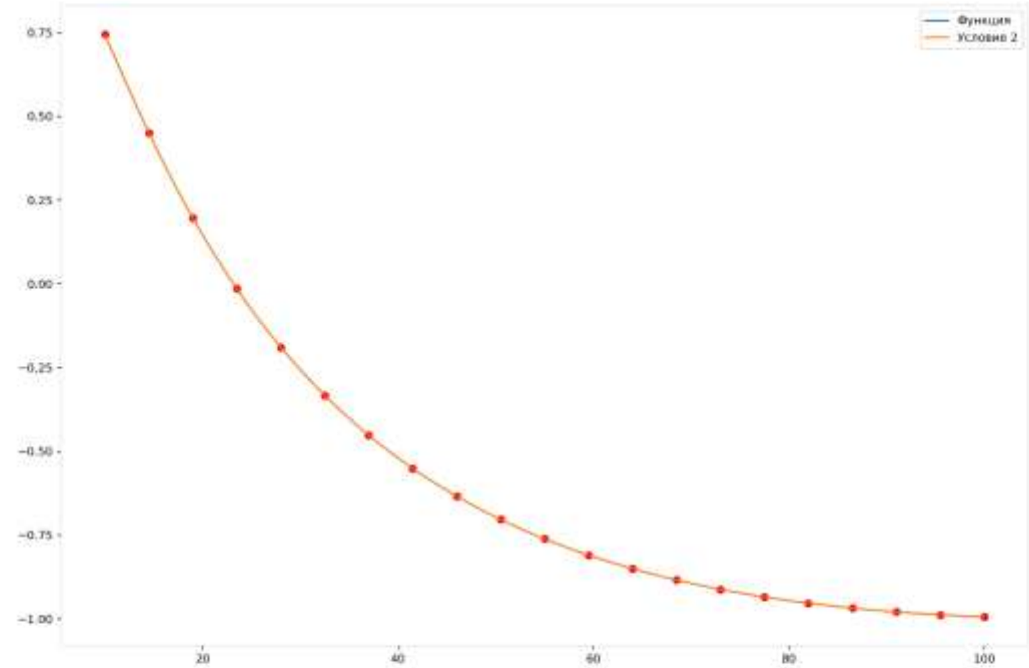
```python
    plt.plot(x, y, label='Функция')
    plt.plot(xInterpNodes, yInterpNodes, 'or')
    plt.plot(x, y_spline_points, label=f'Условие {kind}')

    print('Разность в узлах интерполирования')
    print(yInterpNodes - spline(xInterpNodes))

    print('Max:')
    h = (b_-a_)/n
    middle_points = xInterpNodes[:-1] + h/2
    max_dev = np.max(np.abs(vfunc(middle_points) - spline(middle_points)))
    print(f'{max_dev:.12f}')

    plt.legend()
    save_plt(plt)
    plt.show()


def common_plot():
    plt.figure(num=None, figsize=(15, 10), dpi=180, facecolor='green',
        edgecolor='blue')
    x = np.linspace(a_, b_, 100)
    y = vfunc(x)
    xInterpNodes = np.linspace(a_, b_, n+1)
    yInterpNodes = vfunc(xInterpNodes)


    spline1 = spline_(a_, b_, n, vfunc, 1)
    _spline1 = spline1(x)
    spline2 = spline_(a_, b_, n, vfunc, 2)
    _spline2 = spline2(x)
    spline3 = spline_(a_, b_, n, vfunc, 3)
    _spline3 = spline3(x)

    plt.plot(x, y, label='f')
    plt.plot(x, _spline1, label='Условие 1')
    plt.plot(x, _spline2, label='Условие 2')
    plt.plot(x, _spline3, label='Условие 3')

    plt.legend()

    save_plt(plt)
    plt.show()


s3(1)
s3(2)
s3(3)
common_plot()
```

Результаты

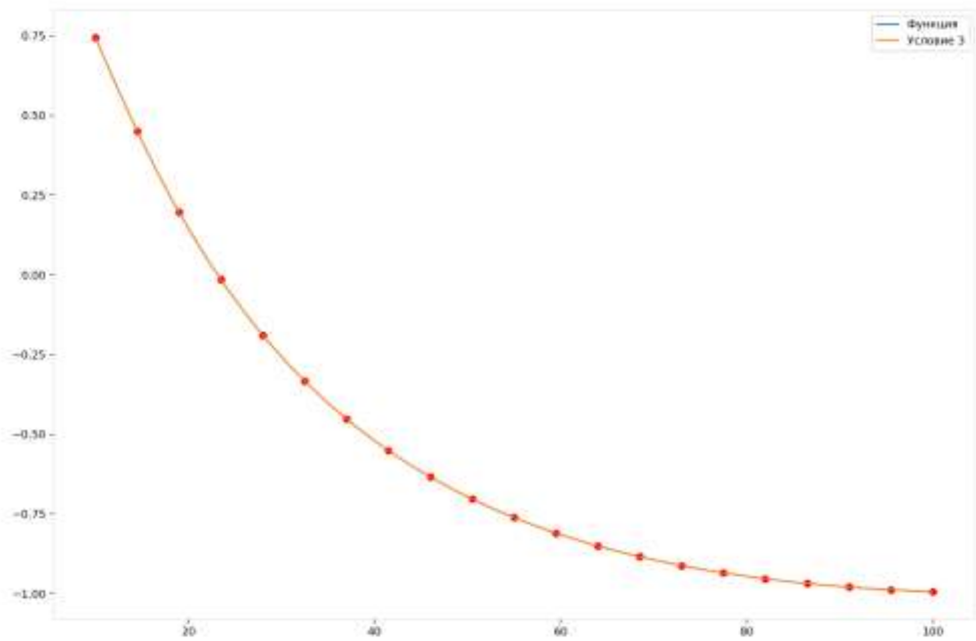
1 доп условие



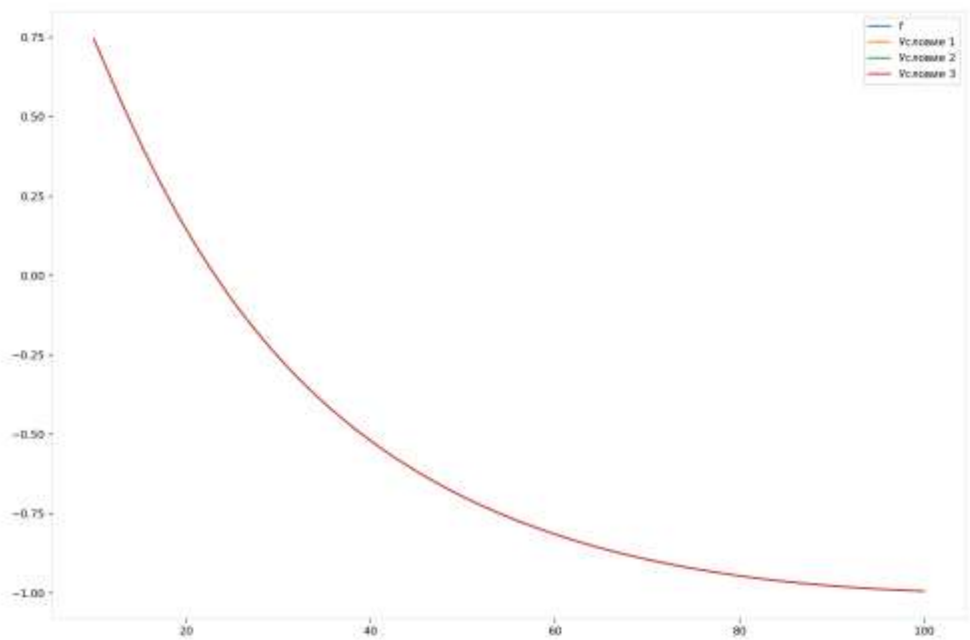
2 доп условие



3 доп условие



Общий график



MAX1: 0.000403594579

MAX2: 0.001025175227

MAX3: 0.000323083031

Вывод:

Метод кубического сплайна пошагово и точно переходит к нужному результату. Данный метод очень прост в использовании, что помогает экономить время, не решая лишних уравнений

Для задания кубического интерполяционного сплайна необходимо определить граничные условия, так как они влияют на результат. Из рассмотренных условий, первое и третье обеспечили наименьшую погрешность.