МИНИСТЕРСТВО НАУКИ И ОБРАЗОВАНИЯ РФ

Федеральное государственное автономное
образовательное учреждение высшего образования
«Национальный исследовательский университет
ИТМО»

ФАКУЛЬТЕТ ПРОГРАММНОЙ ИНЖЕНЕРИИ И КОМПЬЮТЕРНОЙ ТЕХНИКИ

ЛАБОРАТОРНАЯ РАБОТА №1

по дисциплине
ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

Вариант – интерполяция кубическими сплайнами

Выполнил:
Студент группы P3232
Чмурова Мария
Владиславовна

г. Санкт-Петербург
2024 год

Оглавление

Задание

Дан набор точек, по которым необходимо построить интерполяционную функцию по методу кубических сплайнов. Также задана координата x, для которой необходимо найти значение интерполяционного полинома.

Формат входных данных:

x1 x2 x3 ...

y1 y2 y3 ...

x

где x1...xn - список значений аргумента для узлов интерполяции, y1...yn - список значений функции для соответствующего значения аргумента для узлов интерполяции, x - значение аргумента, для которого требуется найти значение интерполяционного полинома. В тестах также вначале задаётся количество узлов интерполяции, однако, в функцию этот параметр не передаётся.

Формат выходных значений: вещественное число, являющееся значением интерполяционной функции в точке x.

Описание метода

По имеющимся координатам точек (узлов интерполяции) необходимо восстановить кривую того, как изменялось значение функции в зависимости от изменения значения аргумента.

Уравнение кубического сплайна на отдельном участке:

$$S_0 = a + b(x - x_0) + c(x - x_0)^2 + d(x - x_0)^3$$

На каждом отдельном интервале необходимо построить различные уравнения сплайнов.

Например, для трех участков (Рис. 1) должно получиться три уравнения:

$$S_0 = a_0 + b_0(x - 1) + c_0(x - 1)^2 + d_0(x - 1)^3$$

$$S_1 = a_1 + b_1(x - 3) + c_1(x - 3)^2 + d_1(x - 3)^3$$

$$S_2 = a_2 + b_2(x - 5) + c_2(x - 5)^2 + d_2(x - 5)^3$$

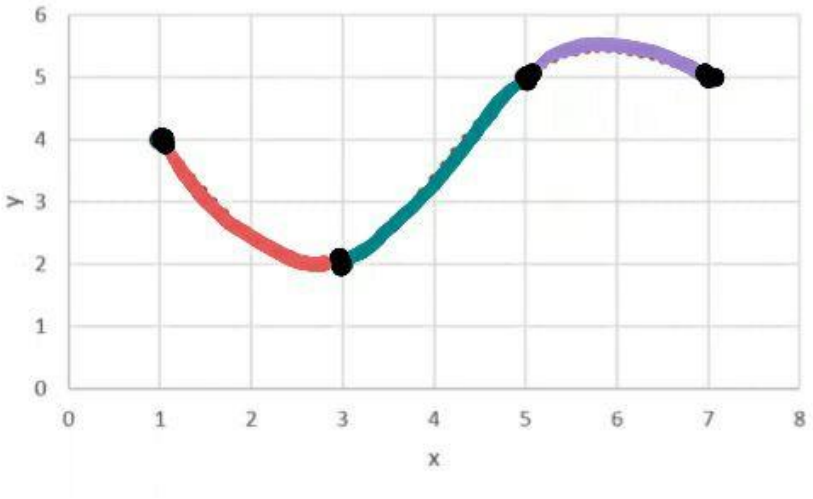


Рис. 1

Для расчета неизвестных коэффициентов используются следующие условия:

1. Сплайны должны проходить через узловые точки. Таким образом, для каждого сплайна получается по два уравнения, в которые подставляются значения его крайних точек.

   Например, для сплайна 1 (Рис. 1):

   $$4 = a_0 + b_0(1 - 1) + c_0(1 - 1)^2 + d_0(1 - 1)^3$$

   $$2 = a_0 + b_0(3 - 1) + c_0(3 - 1)^2 + d_0(3 - 1)^3$$

   Аналогично для остальных двух сплайнов.

Таким образом, имеется $((n - 1) \cdot 2)$ уравнений (где n – количество узлов интерполяции)

2. В стыках сплайнов (в точках соединения) должна обеспечиваться гладкость – не должно быть изломов и изменения кривизны.
   Для этого необходимо приравнять следующие условия: то, под каким углом сплайн заходит в определенную точку (1-ая производная) и то, какую кривизну имеет сплайн в определенной точке (2-ая производная)
   Например, для точки соединения сплайна 1 и сплайна 2 (Рис. 1):
   $S_0' = S_1'$
   $S_0'' = S_1''$
   Аналогично для точки соединения сплайна 2 и сплайна 3
   Таким образом, имеется $(n - 2) \cdot 2$ уравнений (где n – количество узлов интерполяции.
3. Для оставшихся уравнений задается поведение сплайнов на концах интервалов (в начальной и конечной точке). Для этого задается нулевая кривизна сплайна (2-я производная равна нулю)
   Например, для начальной и конечной точки соответственно (Рис. 1):
   $S_0'' = 0$
   $S_2'' = 0$
   Таким образом, имеется 2 уравнения.

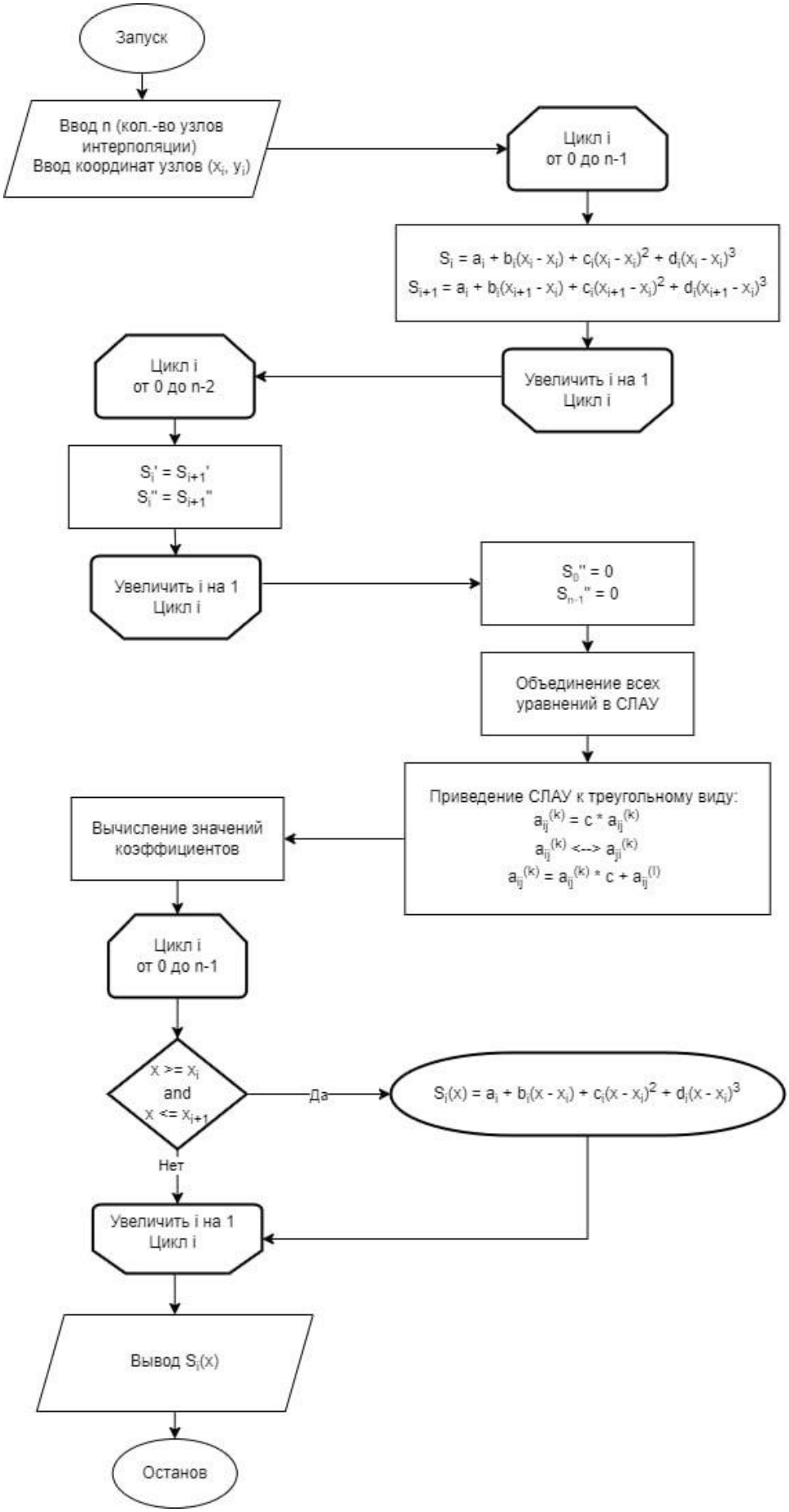
Имеющиеся n уравнений (где n-количество узлов интерполяции) решаются с использованием метода Гаусса:

- Матрица приводится к треугольному виду
- Высчитываются значения коэффициентов

После получения значений всех коэффициентов имеем уравнения кубических сплайнов для каждого участка. По значению X (значение аргумента, для которого требуется найти значение интерполяционного полинома) определяем уравнение кубического сплайна, для которого X

находится внутри его области значений и высчитываем значение функции для этого значения X.

Блок схема


Запуск
Ввод n (кол.-во узлов интерполяции)
Ввод координат узлов (x_i, y_i)
Цикл i от 0 до n-1
$S_i = a_i + b_i(x_i - x_i) + c_i(x_i - x_i)^2 + d_i(x_i - x_i)^3$
$S_{i+1} = a_i + b_i(x_{i+1} - x_i) + c_i(x_{i+1} - x_i)^2 + d_i(x_{i+1} - x_i)^3$
Увеличить i на 1
Цикл i
Цикл i от 0 до n-2
$S_i' = S_{i+1}'$
$S_i'' = S_{i+1}''$
Увеличить i на 1
Цикл i
$S_0'' = 0$
$S_{n-1}'' = 0$
Объединение всех уравнений в СЛАУ
Приведение СЛАУ к треугольному виду:
$a_{ij}^{(k)} = c * a_{ij}^{(k)}$
$a_{ij}^{(k)} <--> a_{ji}^{(k)}$
$a_{ij}^{(k)} = a_{ij}^{(k)} * c + a_{ij}^{(l)}$
Вычисление значений коэффициентов
Цикл i от 0 до n-1
x >= x_i and x <= x_{i+1}
Да
$S_i(x) = a_i + b_i(x - x_i) + c_i(x - x_i)^2 + d_i(x - x_i)^3$
Нет
Увеличить i на 1
Цикл i
Вывод $S_i(x)$
Останов

Код численного метода

```
def interpolate_by_spline(x_axis, y_axis, x):

    length = len(x_axis)

    #создание массива массивов
    inner_length = (length - 1) * 4 + 1
    array_of_arrays = [[] for _ in range(inner_length - 1)]
    for i in range(inner_length - 1):
        array_of_arrays[i] = [0] * inner_length

    #условие 1: сплайны должно проходить через угловые точки

    #определяет x
    x_definition = 0

    #определяет x0 (x1, x2, ..., xn)
    x_i_definition = 0

    #перебирает все уравнения слау по порядку (только половину)
    for i in range((length - 1) * 2):

        #определяем какие коэффициенты заполнять
        if (i % 2 == 0):
            x_definition = int(i / 2)
        else:
            x_i_definition += 1

        #заполняем коэффициенты
        array_of_arrays[i][4 * x_definition + 0] = 1.0
        for j in range(1, 4):
            array_of_arrays[i][4 * x_definition + j] =
(x_axis[x_i_definition] - x_axis[x_definition]) ** j
        array_of_arrays[i][inner_length - 1] =
y_axis[x_i_definition]

    #условие 2: в стыках сплайна должна обеспечиваться гладкость

    x_definition = 0

    #заполняем значениями с половины из уравнений кроме двух
последних
    for i in range((length - 1) * 2, inner_length - 1 - 2):

        #первая производная
        if (i % 2 == 0):
            array_of_arrays[i][4 * x_definition + 1] = 1.0
            array_of_arrays[i][4 * x_definition + 2] = 2 *
(x_axis[x_definition + 1] - x_axis[x_definition])
            array_of_arrays[i][4 * x_definition + 3] = 3 *
(x_axis[x_definition + 1] - x_axis[x_definition]) ** 2
```

```
            array_of_arrays[i][4 * (x_definition + 1) + 1] = -
1.0
        #вторая производная
        else:
            array_of_arrays[i][4 * x_definition + 2] = 2.0
            array_of_arrays[i][4 * x_definition + 3] = 6 *
(x_axis[x_definition + 1] - x_axis[x_definition])
            array_of_arrays[i][4 * (x_definition + 1) + 2] = -
2.0
            x_definition += 1

    #условие 3: задаем поведение сплайна в начальной и конечной
точках

    #начальная точка
    array_of_arrays[inner_length - 1 - 2][4 * 0 + 2] = 2.0

    #конечная точка
    array_of_arrays[inner_length - 1 - 1][4 * (length - 2) + 2]
= 2.0
    array_of_arrays[inner_length - 1 - 1][4 * (length - 2) + 3]
= 6 * (x_axis[length - 1] - x_axis[length - 2])

    #решение СЛАУ методом Гаусса
    result_matrix = gaussian_elimination(array_of_arrays)

    return function_value(x_axis, result_matrix, x)

def function_value(x_axis, result_matrix, x):
    for i in range(len(x_axis) - 1):
        if x >= x_axis[i] and x <= x_axis[i + 1]:
            x_0 = x_axis[i]
            return result_matrix[4 * i + 0] + result_matrix[4 *
i + 1] * (x - x_0) + result_matrix[4 * i + 2] * (x - x_0) ** 2 +
result_matrix[4 * i + 3] * (x - x_0) ** 3
    return "Wrong X"

def gaussian_elimination(matrix):

    rows = len(matrix)
    cols = len(matrix[0])

    #строим треугольную матрицу
    for i in range(rows - 1):

        max_value = 0
        max_row = i

        #находим максимальный по модулю элемент
        for x in range(i, rows):
            if abs(matrix[x][i]) > max_value:
                max_value = abs(matrix[x][i])
                max_row = x
```

```
        #ставим строку с этим максимальным элементом на первое
место
        matrix[i], matrix[max_row] = matrix[max_row], matrix[i]

        #проходимся по всем строчкам
        for j in range(i + 1, rows):

            #коэффициент на который домножить все элементы
матрицы
            koeficient = matrix[j][i] / matrix[i][i] * (-1)

            #домножаем на коэффициент
            for k in range(i, cols):
                matrix[j][k] += koeficient * matrix[i][k]


    #проход по строчкам матрицы в обратном порядке
    for i in range(rows - 1, -1, -1):
        for j in range(i + 1, rows):
            matrix[i][cols - 1] -= matrix[j][cols - 1] *
matrix[i][j]
        matrix[i][cols-1] /= matrix[i][i]

    return [row[cols - 1] for row in matrix]
```

Примеры работы программы

№ Теста	Входные данные	Выходные данные
1	4 1 2 4 7 2 3 1 4 4	1.0
2	10 1 3 5 9 12 25 29 38 100 150 1 4 2 6 4 8 97 54 35 89 10	6.632645554058451
3	4 1 2 4 7 2 3 1 4 10	Wrong X
4	5 1 3 6 9 10 4 7 9 4 2 1	4.0
5	5 1 3 6 9 10 4 7 9 4 2 10	2.0

Вывод

В ходе данной лабораторной работы была разработана программа интерполяции кубических сплайнов, которая успешно интерполирует кривую по заданным координатам узловых точек, благодаря чему получаются уравнения кубических сплайнов для каждого интервала. Кроме того, программа высчитывает значение функции в определенной точке.

Данная программа предотвращает изломы и обеспечивает гладкость и непрерывность в точках соединения уравнений кубических сплайнов.

Данный метод подходит для использования с функциями, имеющими непрерывные производные, так как они необходимы для обеспечивания гладкости кривой.

Алгоритмическая сложность данного метода оценивается с помощью Big O Notation как $O(n^3)$. Данную сложность вызывают внутренние циклы при решении СЛАУ методом Гаусса.