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

**Факультет программной инженерии и компьютерной техники**

**Вычислительная математика**

**Лабораторная работа №1.**

**Интерполирование кубическими сплайнами**

**Выполнил:**

**Маликов Глеб Игоревич**

**Группа № P3224**

**Преподаватели:**

**Перл Ольга Вячеславовна**

**Хохлов Александр Алексеевич**

**г. Санкт-Петербург**

**2024**

Оглавление

[Задание 3](#_Toc160914269)

[Описание численного метода 4](#_Toc160914270)

[Полиномы Чебышёва 4](#_Toc160914271)

[Узлы Чебышёва 4](#_Toc160914272)

[Кубические сплайны 4](#_Toc160914273)

[Блок-схемы 6](#_Toc160914274)

[Код 9](#_Toc160914275)

[cubic\_spline\_interpolation 9](#_Toc160914276)

[chebyshev\_nodes 10](#_Toc160914277)

[interpolate\_by\_spline 11](#_Toc160914278)

[Пример работы программы 13](#_Toc160914279)

[Пример 1 13](#_Toc160914280)

[Пример 2 13](#_Toc160914281)

[Пример 3 13](#_Toc160914282)

[Пример 4 14](#_Toc160914283)

[Пример 5 14](#_Toc160914284)

[Пример 6 15](#_Toc160914285)

[Пример 7 15](#_Toc160914286)

[Пример 8 15](#_Toc160914287)

[Пример 9 16](#_Toc160914288)

[Вывод 17](#_Toc160914289)

[Приложения 18](#_Toc160914290)

# Задание

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

Формат входных данных:

f

a

b

x

где f - номер интерполируемой функции, a и b - границы интерполируемого интервала и x - значение аргумента для интерполяционного полинома.

Степень полинома Чебышева (количество узлов интерполяции) следует увеличивать до тех пор, пока модуль разницы между значениями интерполирующей функции в искомой точке x не будет меньше чем 0.01.

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

# Описание численного метода

## Полиномы Чебышёва

Полиномы Чебышёва представляют собой две последовательности полиномов, связанных с функциями косинуса и синуса, обозначаемых как и . В данной работе используются полиномы Чебышёва первой степени, для нахождения узлов интерполяции.

Многочлен Чебышёва первого рода характеризуется как многочлен степени n со старшим коэффициентом , который меньше всего отклоняется от нуля на отрезке [-1,1]. Данные полиномы могут быть определены несколькими способами:

Рекуррентные формулы:

Явные формулы:

Тригонометрическое определение:

## Узлы Чебышёва

Узлы Чебышёва представляют собой определенные вещественные алгебраические числа, используемые в качестве узлов для интерполяции полиномами. Они представляют собой проекции равномерно распределенных точек на единичной окружности на вещественный интервал [−1, 1], диаметр которой равен 1.

Узлы Чебышёва первого рода, также известные как нули полиномов Чебышёва первого рода, являются корнями этих полиномов. Полиномиальные интерполянты, построенные на основе этих узлов, минимизируют влияние эффекта Рунге.

Формула для нахождения узлов Чебышёва на интервале [a, b] определена следующим образом:

## Кубические сплайны

Сплайны представляют собой гладкие кусочно-полиномиальные функции, используемые для интерполяции данных. Они разбивают область определения на участки и на каждом участке аппроксимируют функцию полиномом низкой степени. Эти кусочные полиномы соединяются так, чтобы обеспечить гладкость и непрерывность в указанных точках, что делает сплайны эффективными для моделирования кривых или поверхностей при работе с реальными данными.

Кубический сплайн — это гладкая функция, область определения которой разбита на конечное число отрезков, на каждом из которых она совпадает с некоторым кубическим многочленом.

Таким образом, для функции на отрезке , функция определяется как интерполирующая функция для если:

* на каждом отрезке , является многочленом степени не выше третьей;
* имеет непрерывные первую и вторую производные на всём отрезке ;
* в точках выполняется равенство , т. е. сплайн интерполирует функцию в точках .

В данной работе применяется условие «естественного сплайна» для однозначного построения сплайна.

Каждый кубический сплайн для каждого отрезка , определяется как:

Для построения этих сплайнов требуется определить коэффициенты

Эти коэффициенты определяются с помощью следующих условий:

Алгоритм вычисления кубических сплайнов приведён в блок схемах ниже.

# Блок-схемы

Обратите внимание что в блок схемах списки начинаются с 1

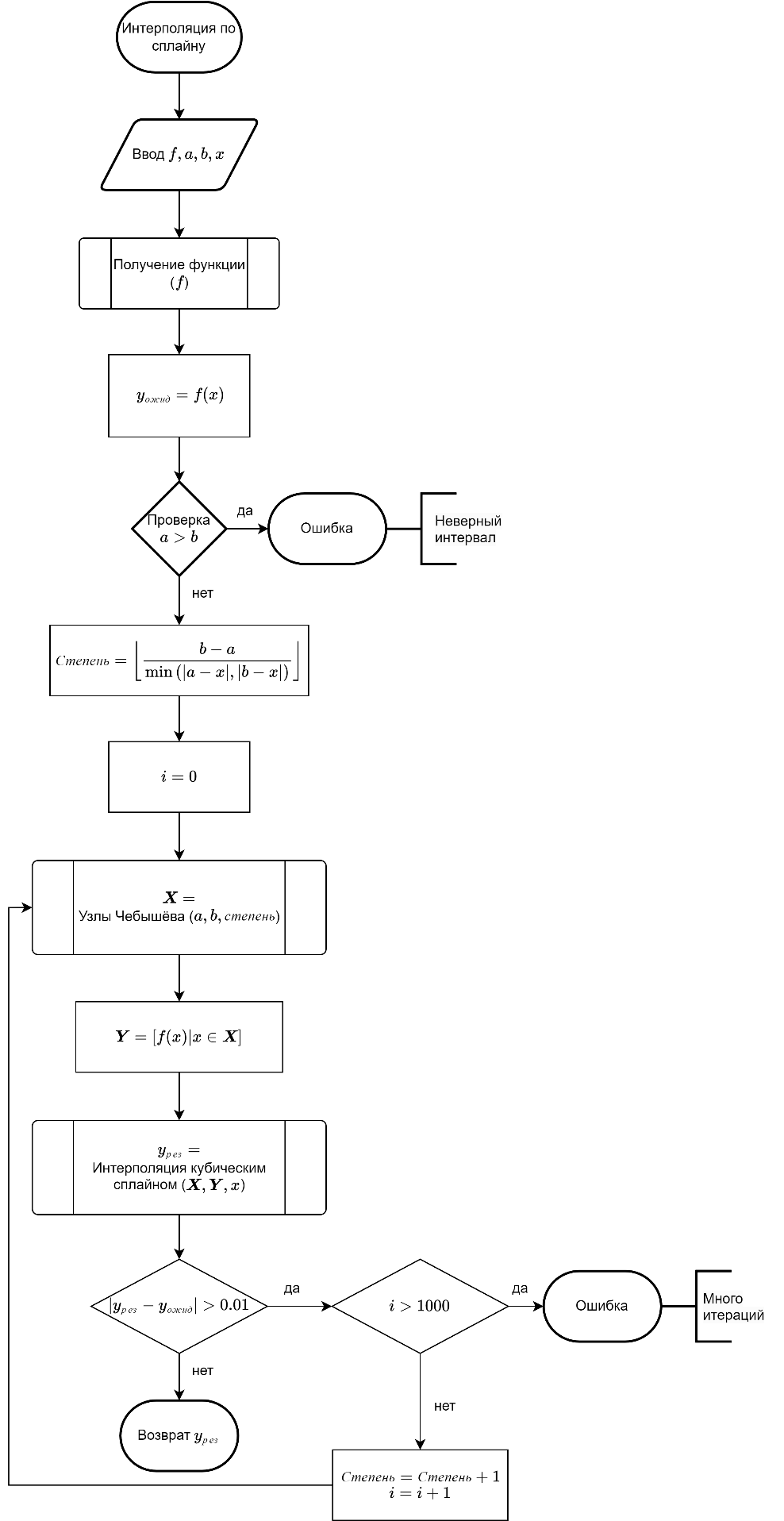


Схема 1 - Интерполяция по сплайну

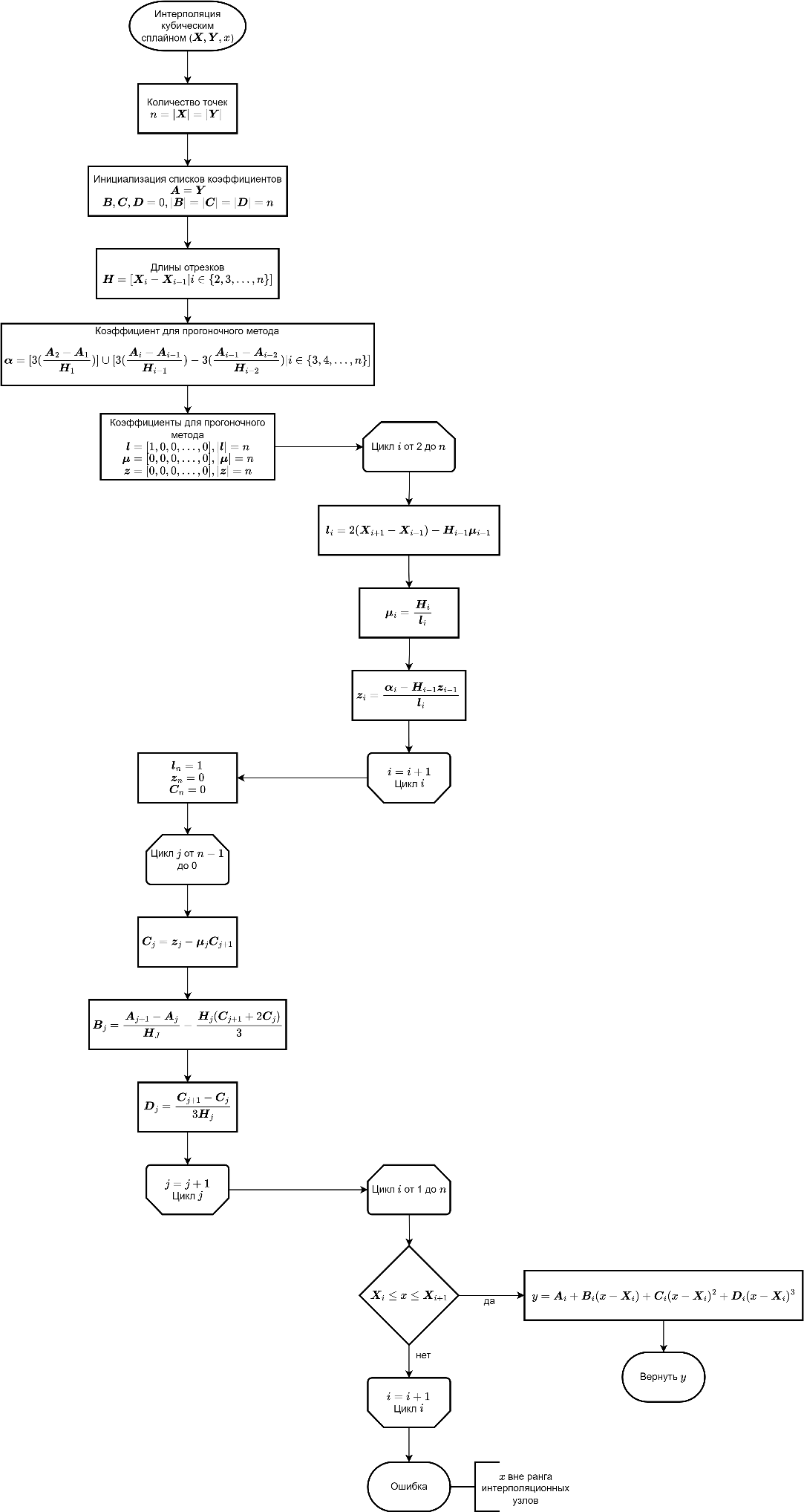


Схема 2 - Интерполяция кубического сплайна

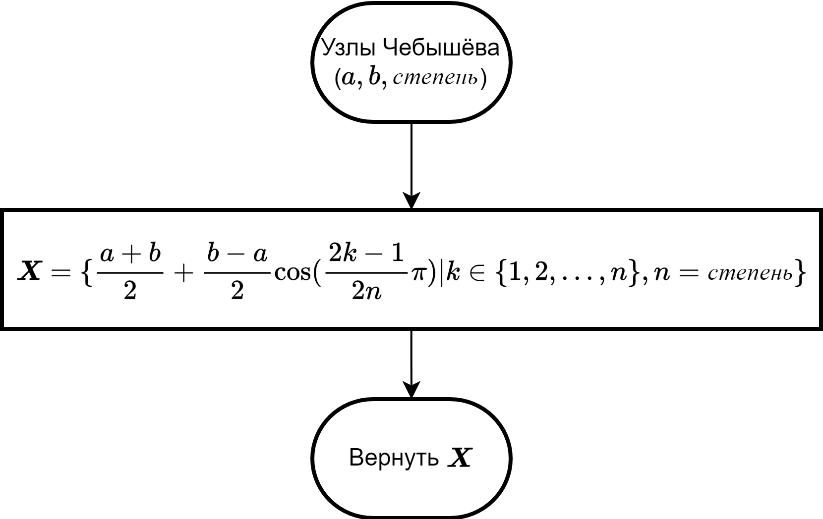


Схема 3 - Узлы Чебышёва

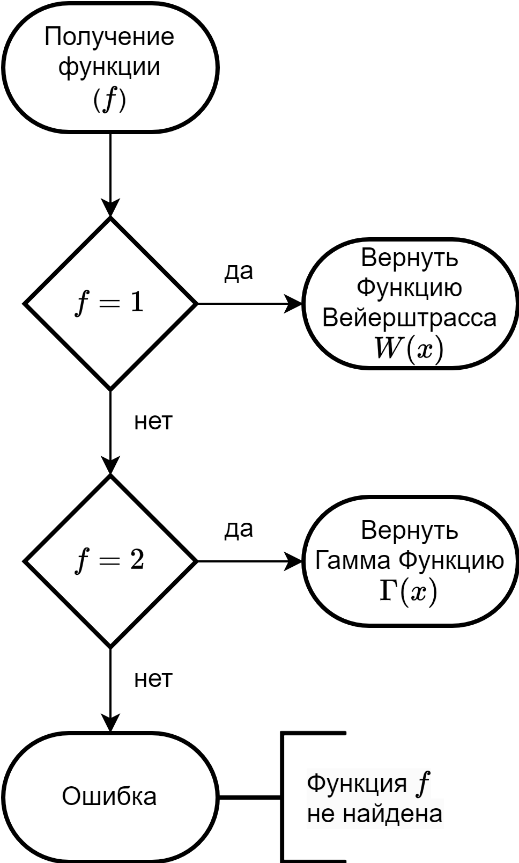


Схема 4 - Получение Функции

# Код

## cubic\_spline\_interpolation

**def** **cubic\_spline\_interpolation**(x\_values, y\_values, x):

"""

Находит значение интерполяционного многочлена в точке x с помощью натурального кубического сплайна

На каждом отрезке [x[i], x[i+1]] интерполяционный многочлен имеет вид:

S[i](x) = a[i] + b[i](x - x[i]) + c[i](x - x[i])^2 + d[i](x - x[i])^3,

где a[i], b[i], c[i], d[i] - неизвестные коэффициенты

Производные интерполяционного многочлена в узлах интерполяции:

S[i](x[i]) = a[i]; S[i]'(x[i]) = b[i]; S[i]''(x[i]) = 2c[i]; S[i]'''(x[i]) = 6d[i],

для всех i = 1, 2, ..., n

Условия непрерывности первой и второй производных интерполяционного многочлена в узлах интерполяции:

S[i](x[i-1]) = S[i-1](x[i-1]) для всех i = 2, 3, ..., n

S[i]'(x[i-1]) = S[i-1]'(x[i-1]) для всех i = 2, 3, ..., n

S[i]''(x[i-1]) = S[i-1]''(x[i-1]) для всех i = 2, 3, ..., n

:param x\_values: Узлы интерполяции

:param y\_values: Значения функции в узлах интерполяции

:param x: Точка, в которой необходимо найти значение интерполяционного многочлена

:return: Значение интерполяционного многочлена в точке x

"""

# Количество узлов интерполяции

n = len(x\_values)

# Коэффициенты интерполяционного многочлена a равны значениям функции в узлах интерполяции

a = y\_values.copy()

# Коэффициенты интерполяционного многочлена b, c, d, которые необходимо найти

b = [**0**] \* n

c = [**0**] \* n

d = [**0**] \* n

# h - длины отрезков [x[i], x[i+1]]

h = [x\_values[i] - x\_values[i - **1**] **for** i **in** range(**1**, n)]

# alpha - коэффициенты для прогоночного метода

alpha = [**3** \* (a[i] - a[i - **1**]) / h[i - **1**] - **3** \* (a[i - **1**] - a[i - **2**]) / h[i - **2**] **for** i **in** range(**2**, n)]

alpha.insert(**0**, **3** \* (a[**1**] - a[**0**]) / h[**0**])

# l, mu, z - Коэффициенты для прогоночного метода

l = [**1**] + [**0**] \* (n - **1**)

mu = [**0**] \* n

z = [**0**] \* n

# Решение прогоночной системы для коэффициентов c

**for** i **in** range(**1**, n - **1**):

l[i] = **2** \* (x\_values[i + **1**] - x\_values[i - **1**]) - h[i - **1**] \* mu[i - **1**]

mu[i] = h[i] / l[i]

z[i] = (alpha[i] - h[i - **1**] \* z[i - **1**]) / l[i]

# Установка граничных условий для второй производной равной нулю в начале и в конце отрезка

l[n - **1**] = **1**

z[n - **1**] = **0**

c[n - **1**] = **0**

# Находим коэффициенты c, b, d

**for** j **in** range(n - **2**, -**1**, -**1**):

c[j] = z[j] - mu[j] \* c[j + **1**]

b[j] = (a[j + **1**] - a[j]) / h[j] - h[j] \* (c[j + **1**] + **2** \* c[j]) / **3**

d[j] = (c[j + **1**] - c[j]) / (**3** \* h[j])

# Находим значение интерполяционного многочлена в точке x если x принадлежит отрезку [x[i], x[i+1]]

**for** i **in** range(n - **1**):

**if** x\_values[i] <= x <= x\_values[i + **1**]:

delta\_x = x - x\_values[i]

**return** a[i] + b[i] \* delta\_x + c[i] \* delta\_x \*\* **2** + d[i] \* delta\_x \*\* **3**

**raise** **ValueError**("x is not in the interval of interpolation nodes")

## chebyshev\_nodes

**def** **chebyshev\_nodes**(a, b, n):

"""

Возвращает узлы интерполяции - корни полиномов Чебышева

:param a: Начало интервала

:param b: Конец интервала

:param n: Количество узлов интерполяции (степень полинома Чебышева)

:return: Список узлов интерполяции

"""

**return** [(a + b) / **2** + (b - a) / **2** \* math.cos(((**2** \* k - **1**) / n) \* (math.pi / **2**)) **for** k **in** range(n, **0**, -**1**)]

## interpolate\_by\_spline

Данная функция отличается от отправленного из-за добавления logging:

#

# Complete the 'interpolate\_by\_spline' function below.

#

# The function is expected to return a DOUBLE.

# The function accepts following parameters:

# 1. INTEGER f

# 2. DOUBLE a

# 3. DOUBLE b

# 4. DOUBLE x

#

**def** **interpolate\_by\_spline**(f, a, b, x):

"""

Интерполяция кубическими сплайнами

:param f: Номер интерполируемой функции

:param a: Начало интервала для значения x

:param b: Конец интервала для значения x

:param x: Значение аргумента для интерполяционного полинома

:return: Значение интерполяционного полинома в точке x

"""

# Функция, которую необходимо интерполировать

**try**:

function = FunctionSet.get\_function(f)

**except** **NotImplementedError**:

**print**(f"Function {f} not defined.")

**return**

# Ожидаемое значение интерполяционного многочлена в точке x

expected\_value = function(x)

logging.debug(f"Expected y value: {expected\_value}")

# Узлы интерполяции

**if** a > b:

**print**("Invalid interval")

**return**

degree = int((b - a) // min(abs(a - x), abs(b - x)))

logging.debug(f"Initial degree: {degree}")

x\_values = chebyshev\_nodes(a, b, degree)

# Значения функции в узлах интерполяции

y\_values = [function(x) **for** x **in** x\_values]

# Находим значение интерполяционного многочлена в точке x

**try**:

res = cubic\_spline\_interpolation(x\_values, y\_values, x)

**except** **ValueError**:

**print**("x is not in the interval of interpolation nodes")

**return**

logging.debug(f"First time interpolated y value: {res}")

# Увеличиваем количество узлов интерполяции, пока модуль разницы между значениями интерполирующей функции в

# искомой точке x, не будет меньше чем 0.01 и повторяем интерполяцию

counter = **0**

**while** abs(res - expected\_value) > **0.01**:

**if** counter > **1000**:

**raise** **ValueError**("Too many iterations")

degree += **1**

x\_values = chebyshev\_nodes(a, b, degree)

y\_values = [function(x) **for** x **in** x\_values]

res = cubic\_spline\_interpolation(x\_values, y\_values, x)

counter += **1**

logging.debug(f"Final interpolated y value: {res}")

logging.debug(f"Relative error: {abs(res - expected\_value)/expected\_value \* 100}%")

logging.debug(f"Final degree: {degree}")

logging.debug(f"Number of iterations: {counter}")

**return** res

# Пример работы программы

## Пример 1

Ввод:

f = 1

a = -1

b = 1

x = -0

Logging:

Expected y value: 1.9375

Initial degree: 2

First time interpolated y value: 1.5606704121123278

Final interpolated y value: 1.9375000000000002

Relative error: 1.1460366705808067e-14%

Final degree: 3

Number of iterations: 1

Вывод:

1.9375000000000002

## Пример 2

Ввод:

f = 1

a = 45

b = 32000

x = 10223

Logging:

Expected y value: -1.9374999999992752

Initial degree: 3

First time interpolated y value: -0.7484846707578217

Final interpolated y value: -1.9287272576473096

Relative error: -0.4527867020371076%

Final degree: 211

Number of iterations: 208

Вывод:

-1.9287272576473096

## Пример 3

Ввод:

f = 2

a = 5

b = 32

x = 30

Logging:

Expected y value: 8.841761992020999e+30

Initial degree: 13

First time interpolated y value: -5.231745055223906e+32

Вывод:

ValueError: Too many iterations

## Пример 4

Ввод:

f = 2

a = 1

b = 15

x = 14

Logging:

Expected y value: 6227020798.650319

Initial degree: 14

First time interpolated y value: 5588839817.169129

Final interpolated y value: 6227020798.646442

Relative error: 6.225587197382835e-11%

Final degree: 659

Number of iterations: 645

Вывод:

6227020798.646442

## Пример 5

Ввод:

f = 1

a = -6

b = 7.3

x = 7.7

Logging:

Expected y value: 1.8426720002780885

Initial degree: 33

Вывод:

x is not in the interval of interpolation nodes

## Пример 6

Ввод:

f = 1

a = -0.99

b = -3.2

x = -2.3

Logging:

Expected y value: 1.8426720002987265

Вывод:

Invalid interval

## Пример 7

Ввод:

f = 1

a = -10.99

b = 10.32

x = -10.1

Logging:

Expected y value: -1.1388339267730325

Initial degree: 23

First time interpolated y value: -1.48705967099345

Final interpolated y value: -1.1330143297926125

Relative error: -0.5110136643812719%

Final degree: 739

Number of iterations: 716

Вывод:

-1.1330143297926125

## Пример 8

Ввод:

f = 2

a = 1

b = 50

x = 2

Logging:

Expected y value: 0.9999999999142317

Initial degree: 49

First time interpolated y value: 3.6343406898069644e+35

Final interpolated y value: 1.0045974133015603

Relative error: 0.4597413387722865%

Final degree: 122

Number of iterations: 73

Вывод:

1.0045974133015603

## Пример 9

Ввод:

f = 3 #Была добавлена функция x^2 для тестирования

a = -560

b = 50

x = -260

Logging:

Expected y value: 67600

Initial degree: 2

First time interpolated y value: 114087.5

Final interpolated y value: 67599.99201135479

Relative error: 1.1817522507302804e-05%

Final degree: 13

Number of iterations: 11

Вывод:

67599.99201135479

# Вывод

Интерполяция кубическими сплайнами показала себя как подходящий способ получения достаточно точной интерполяционной функции, это подтверждается примерами работы программы на различных интервалах и значениях. Подобным образом, использование узлов Чебышёва позволило сократить ошибки интерполяционной функции около пределов интервалов.

Тем не менее, есть несколько случаев, когда данный метод не является оптимальным. Алгоритмическая сложность в этой реализации составляет , где – количество узлов интерполяции, так как составляется один кубический полином для каждого отрезка между узлами. Так как точность интерполяции напрямую зависит от количества узлов, то время выполнения программы увеличивается, и как в примере 3, количество итерации превысило 1000 раз.

Количество узлов зависит от того насколько близко находится значение x от границ интервала, где чем ближе x к границе, тем больше потребуется узлов для уменьшения влияния феномена Рунге. Это может привести к необходимости использовать большое количество узлов, что, в свою очередь, увеличивает вычислительную сложность.

В целом, интерполяция кубическими сплайнами является мощным инструментом для интерполяции функций, но его эффективность и применимость могут варьироваться в зависимости от конкретного сценария использования. Ошибка метода может быть уменьшена, но это может потребовать больше времени на выполнение программы, что является компромиссом, который необходимо учитывать.

# Приложения