МОСКОВСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
имени М.В.Ломоносова
Факультет вычислительной математики и кибернетики

Компьютерный практикум по учебному курсу
«ВВЕДЕНИЕ В ЧИСЛЕННЫЕ МЕТОДЫ»

Численные методы интерполирования и приближения функций

ОТЧЁТ
о выполненном задании

студента 209 учебной группы факультета ВМК МГУ
Кольцова Кирилла Евгеньевича

Содержание

1 Постановка задачи

Задачей является построение интерполянтов разных типов заданных функий. Требуется построить полином Лагранжа для следующих функций $f_i(x)$ на отрезке $x \in [0, 2]$:

$$f_1(x) = \frac{1}{2^5 \cdot 5!} \frac{d^5}{dx^5} \left[\left(x^2 - 1\right)^5 \right],$$

$$f_2(x) = |\sin(4x)|\, e^{2x}.$$

В качестве узлов интерполяции необходимо использовать узлы равномерной на $[0, 2]$ сетки для количества узлов $n = 3, 5, 9, 17$.

Также нужно исследовать сходимость интерполяции и найти максимальные отклонения

$$\max |L_n(x) - f_i(x)|$$

на равномерной сетке из 1001 узла, где $L_n(x)$ - интерполянт. По пути построить графики исходных функций и их интерполянтов, подобрать более эффективный численный метод приближения функции для второй функции.

2 Методы решения

2.1 Метод интерполяции Лагранжа

2.1.1 Описание

Метод интерполяции Лагранжа позволяет построить полином, проходящий через заданные узлы интерполяции. Пусть заданы узлы:

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n),$$

где $y_i = f(x_i)$. Интерполяционный полином Лагранжа имеет вид:

$$L_n(x) = \sum_{i=0}^{n} y_i \phi_i(x),$$

где

$$\phi_i(x) = c_i \prod_{\substack{j=0 \\ j \neq i}}^{n} (x - x_j), c_i = \frac{1}{\prod_{\substack{j=0 \\ j \neq i}}^{n} (x_i - x_j)}, i = 0..n.$$

Построенный многочлен проходит через все узлы интерполирования:

$$f(x_i) = \sum_{k=0}^{n} f(x_k) \phi_k(x_i), i = 0..n.$$

Интерполяция Лагранжа применяется в астрономии для вычисления положения небесных тел и в инженерии для построения калибровочных кривых.

2.1.2 Проблемы вычислений

Основные проблемы данного метода кроются в его основе — многочлене с высокой степенью.

При увеличении числа узлов интерполяции многочлен Лагранжа начинает вести себя нестабильно на краях интервала, что получило особое название — эффект Рунге.

Также для высоких степеней многочлена, возникающих при увеличении числа узлов, расчёты становятся численно неустойчивыми из-за ошибок округления.

Кроме того, если придерживаться определения, то при добавлении новых точек необходимо пересчитывать весь многочлен, что делает метод малоэффективным для задач с динамическим изменением данных.

2.1.3 Барицентрическая форма

Барицентрическая форма полинома Лагранжа призвана решить некоторые проблемы, а именно пересчет многочлена при добавлении точки и численную неустойчивость.

Заметим, что если мы возьмём в качестве функции x^m и посчитаем значения x_i^m в узлах интерполирования, то получим формулу:

$$x^m = \sum_{i=0}^{n} x_i^m \cdot \phi_i(x).$$

Дифференцируя это тождество по x, будем иметь для $m = 0..n$:

$$\sum_{i=0}^{n} x_i^m \phi_i(x) = m x^{m-1},$$

из которой при $m = 0$ вытекает

$$\sum_{i=0}^{n} \phi_i(x) = 1.$$

Далее, если ввести полином $\omega_{n+1}(x) = \prod_{k=0}^{n}(x - x_k)$, то можно записать для $x \neq x_i$

$$\phi_i(x) = c_i \frac{\omega_{n+1}(x)}{x - x_i}.$$

Исходную формулу можно переписать в виде

$$L_n(x) = \omega_{n+1}(x) \sum_{i=0}^{n} \frac{c_i f_i}{x - x_i}.$$

С другой стороны, для $m = 0$ имеем

$$1 = \omega_{n+1}(x) \sum_{i=0}^{n} \frac{c_i}{x - x_i}.$$

Поделив одно на другое, получим при $x \neq x_i$

$$L_n(x) = \frac{\sum_{i=0}^{n} \frac{c_i f_i}{x - x_i}}{\sum_{i=0}^{n} \frac{c_i}{x - x_i}}.$$

Это представление является барицентрической формой. Подобный способ вычисления более экономичен и устойчив к ошибкам округления.

2.1.4 Результаты вычислений

Для $f_1(x)$

Были проведены вычисления интерполянтов по узлам равномерной на $[0, 2]$ сетки для количества узлов $n = 3, 5, 9, 17$ и построены графики функций. Также для каждого интерполянта было посчитано максимальное отклонение на равномерной сетке из 1001 узла. Результаты приведены ниже:

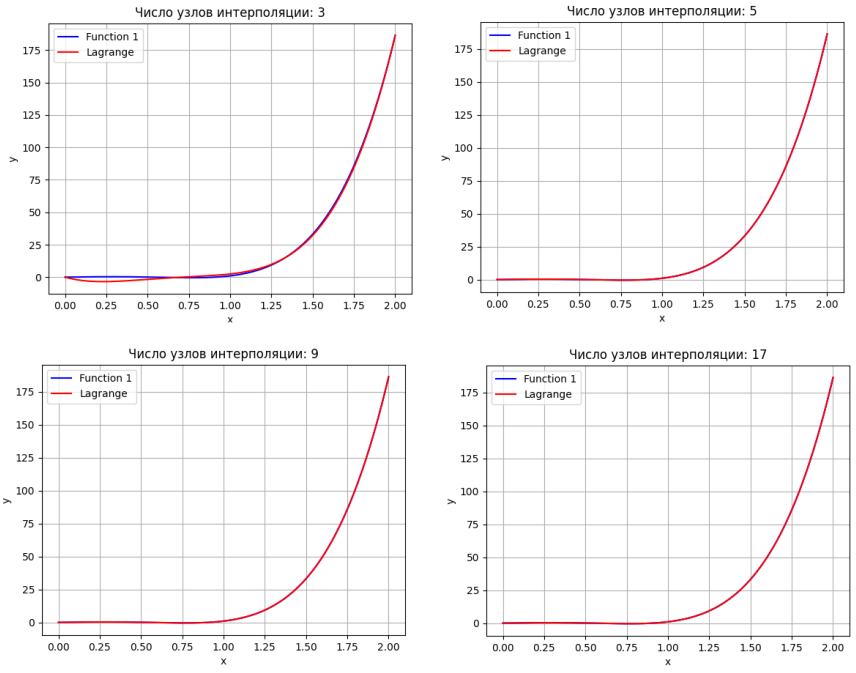

Рис. 1: Графики $f_1(x)$ и полинома лагранжа для количества узлов интерполяции $n = 3, 5, 9, 17$ на $[0, 2]$

Примечательно рассмотреть отрезок $[0, 1]$:

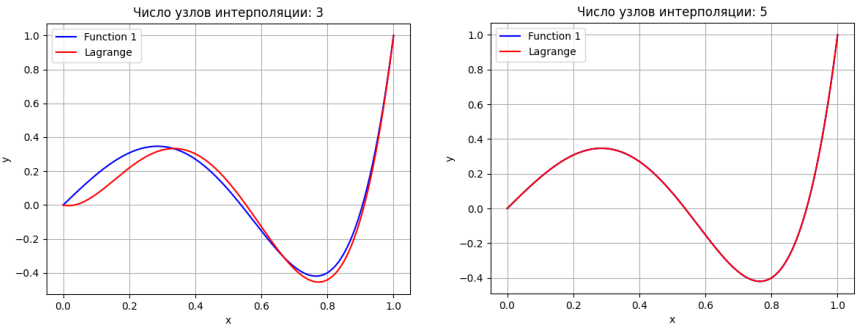

Рис. 2: Графики $f_1(x)$ и полинома лагранжа для количества узлов интерполяции $n = 3, 5$ на $[0, 1]$

Максимальные отклонения для количества узлов интерполяции $n = 3, 5, 9, 17$ на сетке из 1001 узла приведены в таблице:

$n = 3$	$n = 5$	$n = 9$	$n = 17$
3.765905941283157	1.3244516594568267e-11	1.34718902700115e-11	1.3244516594568267e-11

Таблица 1: Отклонения значений полинома Лагранжа от значений $f_1(x)$ в зависимости от количества узлов

Для $f_2(x)$ (аналогично)

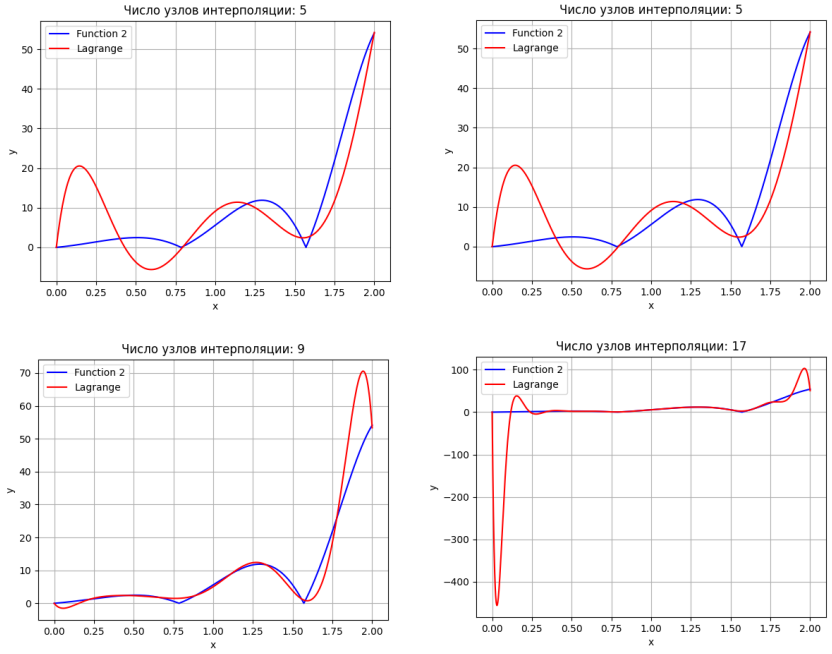

Рис. 3: Графики $f_2(x)$ и полинома Лагранжа для количества узлов интерполяции $n = 3, 5, 9, 17$ на $[0, 2]$

$n = 3$	$n = 5$	$n = 9$	$n = 17$
20.44686836622694	19.829961952477838	22.831417382271724	456.19184772051335

Таблица 2: Отклонения значений полинома Лагранжа от значений $f_2(x)$ в зависимости от количества узлов

2.1.5 Исследование на сходимость

Исследование на сходимость проводилось на сетке из 1001 узла, в каждом из которых сравнивалось значение функции и интреполянта. На графиках в левом столбце показана зависимость отклонения от количества узлов, вплоть до 80 узлов (для наглядности роста). В правом же столбце вплоть до 500. В первой строке исследуется функция $f_1(x)$, во второй — $f_2(x)$.

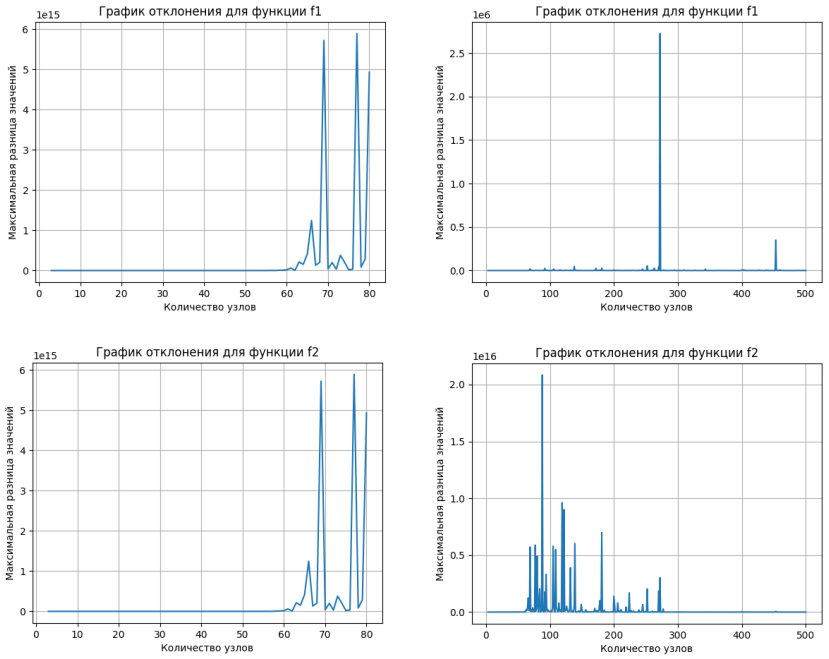

Рис. 4: Исследование полинома Лагранжа на сходимость на $[0, 2]$

Явно виден эффект Рунге, по которому при увеличении количества узлов растет отклонение.

2.2 Метод интерполяции кубическими сплайнами

2.2.1 Описание

Метод интерполяции кубическими сплайнами был выбран неспроста. Метод Лагранжа обаладает некоторыми серьёзными недостатками при интерполяции $f_2(x)$:

- Функция $|sin(4x)|$ имеет резкие углы и изменения на некоторых участках, что приводит к высокой чувствительности и колебаниям, особенно вблизи точек пересечений и экстремумов.
- Метод расходится.
- Серьёзное несовпадение значений со значениями функции в большистве точек отрезка, в независимости от количества узлов.

Интерполяция кубическими сплайнами лишена этих недостатков. Построение соответствующего интерполянта выглядит следующим образом. На отрезке вводится сетка

$$(x_0, y_0), (x_1, y_1), \ldots, (x_n, y_n),$$

где $y_i = f(x_i)$. *Сплайном*, соответствующим данной функции $f_i(x)$ и данным узлам, называется функция $s(x)$, удовлетворяющая следующим условиям:

1. на каждом сегменте $[x_{i-1}, x_i], i = 1..N$ функция $s(x)$ является многочленом третьей степени;
2. функция $s(x)$, а также ее первая и вторая производные непрерывны на $[a, b]$;
3. $s(x_i) = f(x_i), i = 0..N$

На каждом из отрезков $[x_{i-1}, x_i], i = 1..N$ будем искать функцию $s(x) = s_i(x)$ в виде многочлена третьей степени

$$s_i(x) = a_i + b_i(x - x_i) + \frac{c_i}{2}(x - x_i)^2 + \frac{d_i}{6}(x - x_i)^3,$$
$$x_{i-1} \le x \le x_i, i = 1..N.$$

Из условий интерполирования и непрерывности первой и второй производных получаем

$$a_i = f(x_i), i = 1..N,$$

$$h_i c_{i-1} + 2(h_i + h_{i+1})c_i + h_{i+1}c_{i+1} = 6(\frac{y_{i+1} - y_i}{h_{i+1}} - \frac{y_i - y_{i-1}}{h_i}), i = 1..N-1, c_0 = c_N = 0,$$

$$d_i = \frac{c_i - c_{i-1}}{h_i}, i = 1..N,$$

$$b_i = \frac{h_i}{2}c_i - \frac{h_i^2}{6}d_i + \frac{y_i - y_{i-1}}{h_i}, i = 1..N$$

(граничные значения c_0 и c_N выбраны нулевыми искусственно). Система линейных уравнений для коэффициентов c_i имеет трехдиагональную матрицу, а значит имеет единственное решение. Кроме того, наблюдается диагональное преобладание, что позволяет отыскать решение методом прогонки, которая в данном случае устойчива.

Сплайны, благодаря их плавности и устойчивости, широко используются в компьютерной графике для создания сглаженных кривых, в гидродинамике для моделирования жидкостей, а также в обработке данных и сигналов для заполнения пропусков и сглаживания функций.

2.2.2 Результаты вычислений

Для $f_1(x)$

Результат для первой функции схож с результатом интерполирования методом Лагранжа:

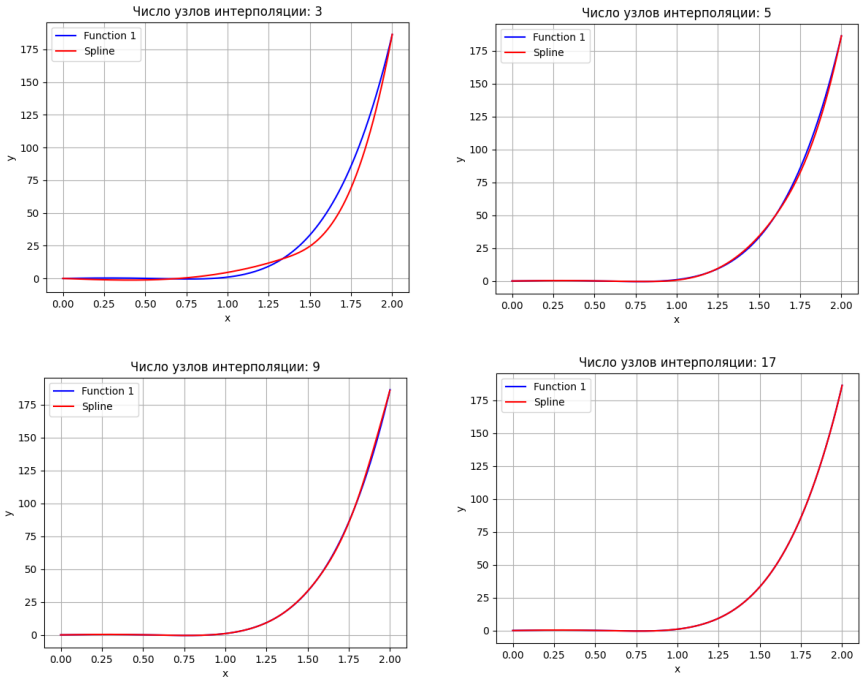

Рис. 5: Графики $f_1(x)$ и кубического сплайна для количества узлов интерполяции $n = 3, 5, 9, 17$ на $[0, 2]$

$n = 3$	$n = 5$	$n = 9$	$n = 17$
16.81075862747487	4.352510044627834	2.7712309500427637	0.25238700112780066

Таблица 3: Отклонения значений сплайна от значений $f_1(x)$ в зависимости от количества узлов

В то же время интерполирование кубическими сплайнами при тех же параметрах дало заметно более точный результат для $f_2(x)$:

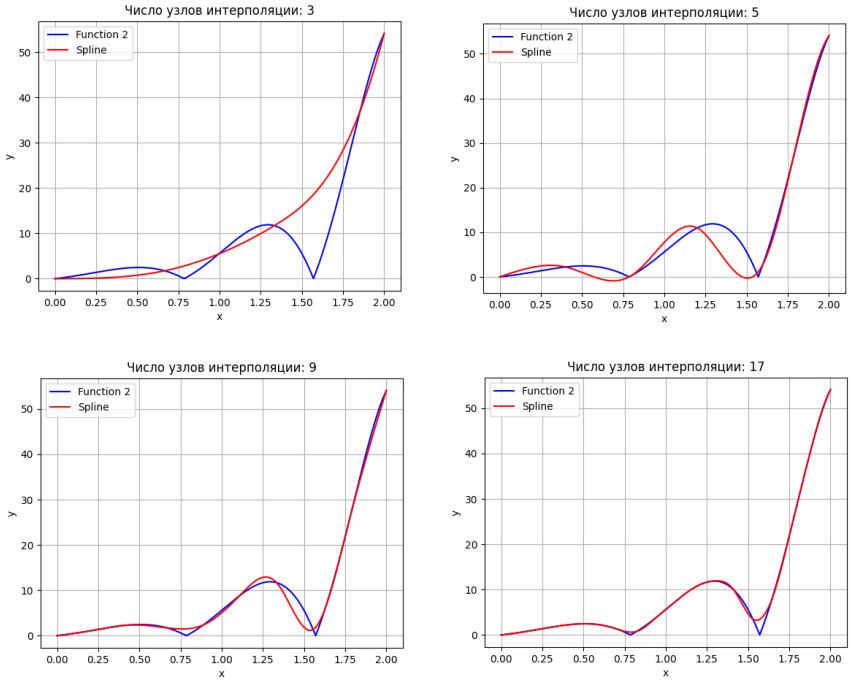

Рис. 6: Графики $f_2(x)$ и кубического сплайна для количества узлов интерполяции $n = 3, 5, 9, 17$ на $[0, 2]$

$n = 3$	$n = 5$	$n = 9$	$n = 17$
18.51850834975229	8.16278232814281	4.166007445350302	3.4108455942021894

Таблица 4: Отклонения значений сплайна от значений $f_2(x)$ в зависимости от количества узлов

2.2.3 Исследование на сходимость

Исследование на сходимость в случае интерполяции кубическими сплайнами проводилось по тому же образцу:

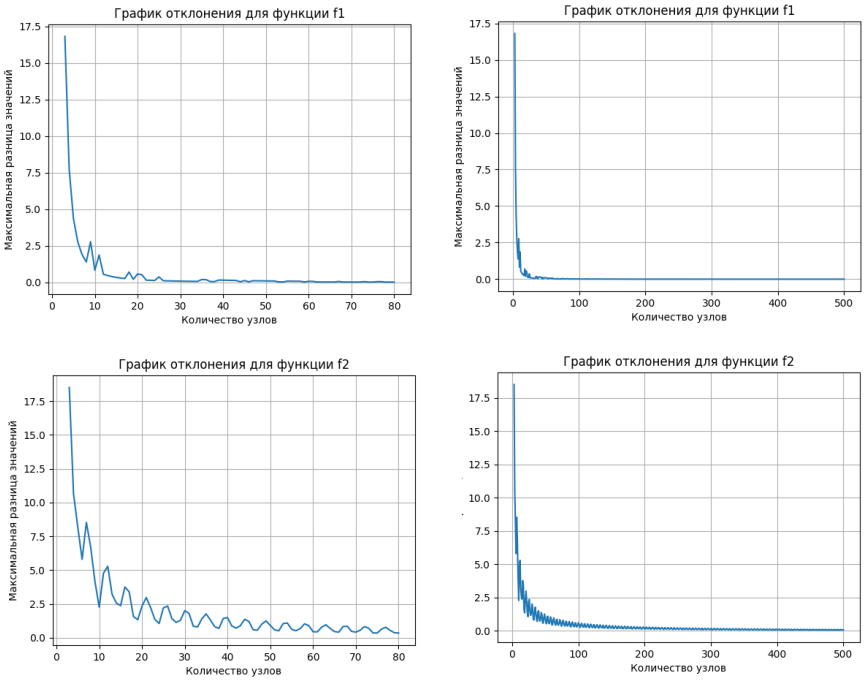

Рис. 7: Исследование сплайна на сходимость на $[0, 2]$

Отчётливо видно, что в обоих случаях метод сходится. Также примечательно посмотреть на график сплайна при очень большом количестве узлов интерполяции, например при 1001:

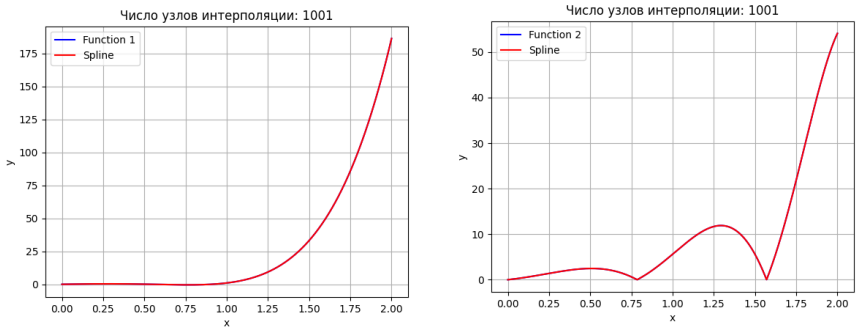

Рис. 8: Графики $f_1(x)$, $f_2(x)$ и их сплайном при большом количестве узлов инерполяции на $[0, 2]$

В обоих случаях график сплайна "закрашивает"график исходной функции, что означает весьма точное дублирование её значений.

3 Программная реализация

Программа, реализующая все компоненты работы написана на языке Python. Для отрисовки графиков функций использовалась библиотека MatPlotLib, для расчёта математических функций e^x и $sin(x)$ использовалась встроенная библиотека math.

Все ключевые параметры настраиваются через глобальные переменные, определенные в самом начале программы.

Оба метода реализованы в виде классов `LagrangeInterpolant` и `SplineInterpolant`.

3.1 Инициализация

Класс содержит в себе две функции: инициализирующую (`__init__`) и возвращающую значение по данному на вход x (`Calc`). При создании объекта любого из классов происходит инициализация:

- В случае с полиномом Лагранжа происходит расчёт барицентрических весов, которые зависят только от узлов и значений исходной функции в них. В описанных выше формулах это коэффициенты c_i:

```
    def __init__(self, nodes, values):
        self.nodes = nodes
        self.values = values
        self.n = len(nodes)

        # Предварительный расчет барицентрических весов
        self.c = [1] * self.n
        for j in range(self.n):
            for k in range(self.n):
                if k != j:
                    self.c[j] /= (self.nodes[j] - self.nodes[k])

```

- В случае со сплайном происходит расчёт коэффициентов для всех отрезков по формулам, в том числе решение трехдиагонального СЛАУ с помощью метода прогонки (функция `RunThroughMethod`, описана в коде работы):

```
    def __init__(self, nodes, values):
        self.nodes = nodes
        self.values = values
        self.N = len(nodes) - 1
        nodes_amount = len(nodes)

        # ГРАНИЧНЫЕ ЗНАЧЕНИЯ КОЭФФИЦИЕНТА c НУЛЕВЫЕ
        c0 = 0
```

```
        cN = 0

        '''
            Все массивы имеют избыточный размер для реализации
            индексации ровно как в формулах.
            Во всех циклах и срезах, где фигурирует N (self.N),
            второй предел увеличен на единицу
            в связи с оссобенностями работы python
        '''
        self.a = [0 for _ in range(nodes_amount)]
        self.b = [0 for _ in range(nodes_amount)]
        self.c = []
        self.d = [0 for _ in range(nodes_amount)]
        h = [0 for _ in range(nodes_amount)]

        # Подсчёт коэффициентов a
        for i in range(len(values)):
            self.a[i] = values[i]

        # Подсчёт коэффициентов c
        for i in range(1, self.N + 1):
            h[i] = nodes[i] - nodes[i - 1]

        temp_h = [0 for _ in range(nodes_amount)]
        temp_f = [0 for _ in range(nodes_amount)]
        for i in range(1, (self.N - 1) + 1):
            temp_h[i] = 2 * (h[i] + h[i + 1])
            temp_f[i] = (6 * ((values[i + 1] - values[i]) / h[i + 1] - (values[i] - values[i - 1]) / h[i]))

        temp_f[1] -= h[1] * c0
        temp_f[self.N-1] -= h[self.N] * cN

        self.c = RunThroughMethod(h[2:(self.N-1)+1],
                                  temp_h[1:(self.N-1)+1],
                                  h[2:(self.N - 1) + 1],
                                  temp_f[1:(self.N-1)+1])
        self.c.insert(0, c0)
        self.c.append(cN)

        # Подсчёт коэффициентов d
        for i in range(1, self.N + 1):
            self.d[i] = (self.c[i] - self.c[i - 1]) / h[i]

        # Подсчёт коэффициентов b
        for i in range(1, self.N + 1):
            self.b[i] = h[i] * self.c[i] / 2
            self.b[i] -= h[i]*h[i] * self.d[i] / 6
            self.b[i] += (values[i] - values[i - 1]) / h[i]
```

3.2 Вычисление значениий

Значения вычисляются по следующим правилам:

- В `LagrangeInterpolant` значение считается по барицентрической формуле. Предварительно происходит проверка двух вещественных чисел на равенство с заданной точностью:

```
    # Подсчёт значения для случайного x
    def Calc(self, x):
        numerator = 0
        denominator = 0

        # Вычисляем числитель и знаменатель барицентрической формулы
        for j in range(self.n):
            # Проверка на совпадение с узлом
            if abs(x - self.nodes[j]) < EPSILON:
                return self.values[j]
```

```
11
12              term = self.c[j] / (x - self.nodes[j])
13              numerator += term * self.values[j]
14              denominator += term
15
16          return numerator / denominator
```

- В `SplineInterpolant` считается значение кубического полинома в заданными коэффициентами:

```
1       # Подсчёт значения для случайного x
2       def Calc(self, x):
3           for i in range(1, self.N + 1):
4               if (self.nodes[i - 1] <= x <= self.nodes[i]):
5                   dx = x - self.nodes[i]
6                   res = self.a[i] + self.b[i] * dx + self.c[i] * dx*dx / 2 + self.d[i] * dx*dx*dx / 6
7                   return res
```

3.3 Дополнительные функции

Также реализованы некоторые вспомогательные функции: рисующая график, создающая сетку, исследующая на сходимость, решающая трехдиагональную СЛАУ.

4 Заключение

4.1 Выводы

Метод интерполирования многочленами Лагранжа удобен для небольшого количества узлов благодаря простоте реализации, однако его применение ограничено из-за эффекта Рунге, численной нестабильности и высокой вычислительной сложности при увеличении числа узлов. Кубические сплайны, напротив, обсепечивают устойчивость к численным ошибкам и удобство работы с большим числом точек, однако требуют решения системы линейных уравнений и выбора граничных условий, что может влиять на точность интерполяции. Таким образом, выбор метода зависит от задачи, требуемой точности и количества узлов.

4.2 Литература

Самарский А.А., Гулин А.В. Численные методы. — М.: Наука. стр 127-140

М.Р.Тимербаев, Численные методы. Приближение функций. Численное интегрирование: учебное пособие — Казанский Федеральный Универсиет: стр. 9-13