Отчет по лабораторной работе № 5 и № 6

Вариант № 17

Чилеше Абрахам

Группа: Б9122-02-03-01сцт

Цели работы

1. Реализовать сплайн с помощью метода моментов
2. Создать таблицу ошибок
3. Построить график зависимости абсолютной ошибки от числа узлов
4. Проанализировать поведение ошибки

данные:

1. Функция: $f(x) = 0.5x^2 + \cos(2x)$
2. Отрезок: [0.6, 1.1]

Ход работа

1. Инициализация функции

Сплайн — это кусочная функция, которая на каждом участке между узлами принимает новые коэффициенты для кубического полинома. Для определения этих коэффициентов нужно получить три массива значений. Для их вычисления используется метод последовательного исключения (метод прогонки).

- Функция
  $f(x) = 0.5x^2 + \cos(2x)$

- Первая производная
  $f'(x) = x - 2\sin(2x)$

- Вторая производная
  $F''(x) = 1 - 4\cos(2x)$

```python
# Function and its derivative
def func(x):
    return 0.5 * x ** 2 + np.cos(2 * x)

def f_deriv_func(x):
    return x - 2 * np.sin(2 * x)

def s_deriv_func(x):
    return 1 - 4 * np.cos(2 * x)
```

```python
def build_spline():
    c_coefficient_matrix = [0] + [h_values[i] / (h_values[i] + h_values[i + 1]) for i in range(0, n - 1)] + [0]
    a_coefficient_matrix = [0] + [h_values[i + 1] / (h_values[i] + h_values[i + 1]) for i in range(0, n - 1)] + [0]
    b_coefficient_matrix = [1] + [2] * (n - 1) + [1]

    rhs = [s_deriv_func(interval[0])] + [
        6 / (h_values[i] + h_values[i + 1]) * ((y_values[i + 1] - y_values[i]) / h_values[i + 1] \
                                               - (y_values[i] - y_values[i - 1]) / h_values[i]) for i in
        range(0, n - 1)] + [s_deriv_func(interval[1])]

    alpha = [-c_coefficient_matrix[0] / b_coefficient_matrix[0]]
    betta = [rhs[0] / b_coefficient_matrix[0]]

    moments = [s_deriv_func(interval[1])]

    for i in range(0, n - 1):
        alpha.append(-c_coefficient_matrix[i] / (alpha[i] * a_coefficient_matrix[i] + b_coefficient_matrix[i]))
        betta.append((rhs[i] - betta[i] * a_coefficient_matrix[i]) / (
                    alpha[i] * a_coefficient_matrix[i] + b_coefficient_matrix[i]))

    for i in range(0, n - 1):
        moments.append(alpha[n - i - 1] * moments[i] + betta[n - i - 1])
    moments.append(s_deriv_func(interval[0]));
    moments = moments[::-1]

    a = moments[:-1:]
    b = [(moments[i + 1] - moments[i]) / h_values[i + 1] for i in range(0, n)]
    c = [(y_values[i + 1] - y_values[i]) / h_values[i + 1] - h_values[i + 1] / 6 * (2 * moments[i] + moments[i + 1]) for
         i in
         range(0, n)]

    return [a, b, c]
```

2. Описание алгоритма сплайновой интерполяции:

1. **Определение матриц коэффициентов:**

- Матрица коэффициентов c вычисляется как отношение длины текущего сегмента к сумме длин текущего и следующего сегментов. Первые и последние элементы этой матрицы равны нулю.
- Матрица коэффициентов a определяется как отношение длины следующего сегмента к сумме длин текущего и следующего сегментов. Начальные и конечные элементы этой матрицы также равны нулю.
- Матрица коэффициентов b имеет единицы в начале и в конце, а все промежуточные элементы равны двум.

2. **Определение правой части уравнения:**

   - Правая часть уравнения включает значения второй производной функции на концах интервалов. Для внутренних узлов она определяется как разность отношений изменений значений функции к длинам интервалов, умноженная на 6.

3. **Инициализация и вычисление массивов α и β:**

   - Массив α инициализируется как отношение начального элемента матрицы c к начальному элементу матрицы b с отрицательным знаком.
   - Массив β инициализируется как отношение начального элемента правой части уравнения к начальному элементу матрицы b.

4. **Вычисление моментов M:**

   - Массив моментов M инициализируется значением второй производной функции на правом конце интервала.

5. **Обход в прямом порядке для вычисления α и β:**

   - Для каждого узла (кроме первого и последнего) массивы α и β обновляются на основе значений матриц коэффициентов a, b, c и правой части уравнения.

6. **Обход в обратном порядке для вычисления моментов M:**

   - Для каждого узла (кроме первого и последнего) массив моментов M обновляется на основе значений массивов α и β.

7. **Вычисление коэффициентов a, b и c:**

   - Коэффициенты a соответствуют всем элементам массива моментов M, кроме последнего.
   - Коэффициенты b определяются как разность соседних элементов массива моментов M, деленная на длину соответствующих интервалов.
   - Коэффициенты c рассчитываются как разность значений функции в соседних узлах, деленная на длину соответствующих интервалов, с корректировкой на основе длин интервалов и значений моментов.

8. **Возврат коэффициентов:**

- Функция возвращает массивы коэффициентов a, b и c, которые используются для построения сплайна.

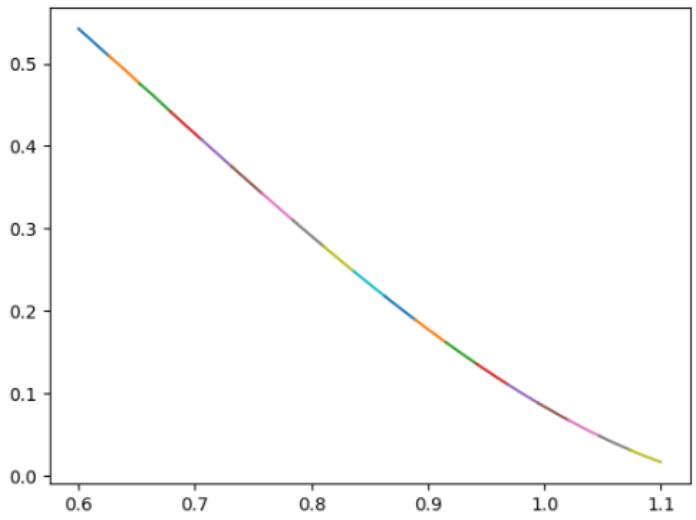
Этот алгоритм гарантирует вычисление необходимых коэффициентов для кубического сплайна с использованием метода прогонки.

Реализация сплайна

```
def evaluate_spline(x, i):
    return y_values[i] + spline_coefficients[2][i] * (x - x_values[i]) + spline_coefficients[0][i] * (
            x - x_values[i]) ** 2 / 2 + spline_coefficients[1][i] * (x - x_values[i]) ** 3 / 6


x_values = np.linspace(*interval, 20)
n = len(x_values) - 1
h_values = [abs(x_values[_] - x_values[_ - 1]) for _ in range(0, n + 1)]
y_values = [func(_) for _ in x_values]

spline_coefficients = build_spline()

for i in range(n):
    xl = np.linspace(x_values[i], x_values[i + 1], 10)
    yl = evaluate_spline(xl, i)

    plt.plot(xl, yl)
```

Алгоритм:

- Создание сетки значений x: Генерируется массив из 20 равномерно распределённых точек x в заданном интервале.
- Вычисление длин сегментов h: Определяются длины каждого сегмента между соседними точками.
- Вычисление значений функции y: Определяются значения функции f(x) в каждой из этих точек.
- Построение коэффициентов сплайна: Вызывается функция build_spline для вычисления коэффициентов сплайна.
- Визуализация сплайна: Для каждого сегмента рассчитываются значения сплайна, и строится график для этого сегмента.



3. Ошибки

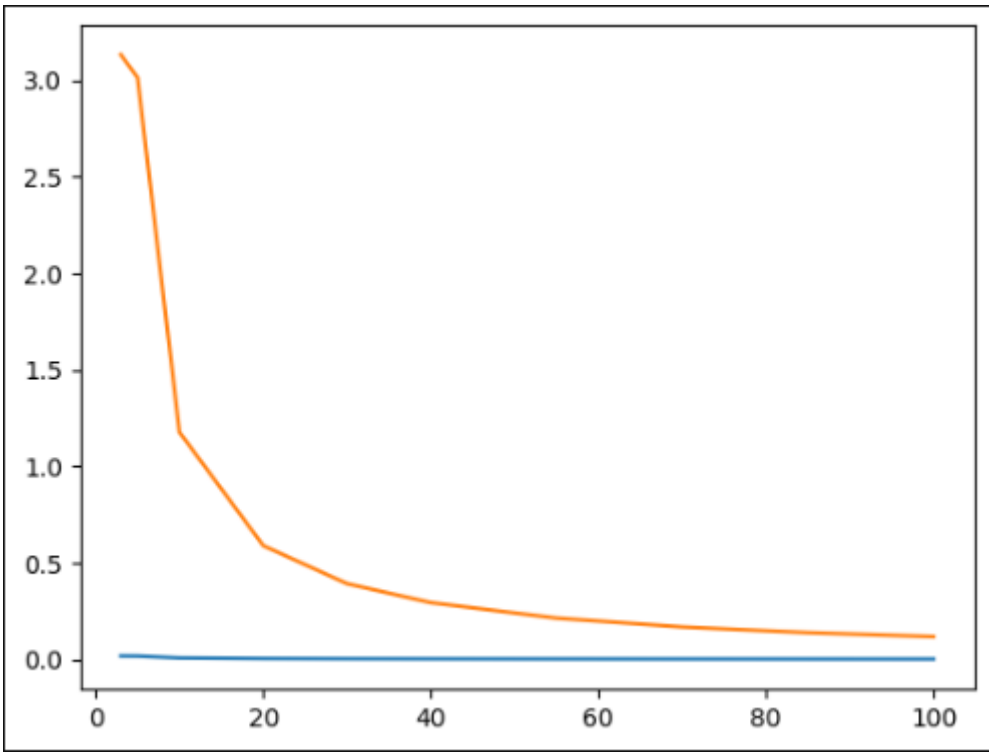
```python
def norm(lst):
    return max(list(map(np.fabs, lst)))


ns = [3, 5, 10, 20, 30, 40, 55, 70, 85, 100]
max_deviations = []
relative_deviations = []

print("=" * 56)

for num_intervals in ns:

    spline_y_values = []
    x_values = np.linspace(*interval, num_intervals)
    n = num_intervals - 1
    h_values = [abs(x_values[_] - x_values[_ - 1]) for _ in range(0, n + 1)]
    y_values = [func(_) for _ in x_values]

    spline_coefficients = build_spline()

    for i in range(n):
        xl = np.linspace(x_values[i], x_values[i + 1], 10)
        yl = evaluate_spline(xl, i)

        spline_y_values += [*yl]

    original_y_values = func(np.linspace(*interval, n * 10))

    spline_norm = norm(np.array(spline_y_values) - original_y_values)
    func_norm = norm(original_y_values)
```

Таблица ошибок

```
========================================================
3       0.016985877122662363    3.1318584425979497
5       0.016335519228396822    3.0119453614448877
10      0.006378458737707482    1.1760611303257056
20      0.003190435108451628    0.5882528796016038
30      0.002122205352131145 7  0.391292525019557
40      0.001589284785914246 7  0.29303255513471244
55      0.001154237782722056 2  0.21281852673717036
70      0.000906120857313119 6  0.16707069269940555
85      0.000745783078170525 5  0.13750759014963085
100     0.000633650195174806 4  0.11683251321560283
```

Таблица: абсолютных и относительных отклонений

График зависимости абсолютной ошибки от количества узлов



Вывод о поведении ошибок

На основе представленных данных можно сделать следующие выводы:

• Сокращение абсолютной ошибки (Δ): При увеличении количества интервалов n абсолютное отклонение (Δ) последовательно уменьшается. Это означает, что при большем числе интервалов аппроксимация сплайном становится точнее.

• Снижение относительной ошибки (δ): Точно так же относительное отклонение (δ) уменьшается с увеличением количества интервалов. Это указывает на то, что относительная ошибка аппроксимации снижается, делая аппроксимацию более точной по сравнению с исходной функцией.

• Темп уменьшения ошибок: Как видно из данных, самое значительное снижение ошибок наблюдается при небольших значениях n. При более высоких значениях n уменьшение ошибок становится менее заметным. Это может означать, что дальнейшее увеличение числа интервалов приводит к менее значительному улучшению точности аппроксимации. В целом, результаты показывают, что увеличение числа интервалов n улучшает точность аппроксимации сплайном как в абсолютном, так и в относительном выражении.