Отчет для программы по численным методам – Худобин В. О.
Задача №2: (№4) Аппроксимация заданной табличной функции y(x) на отрезке и визуализация решения
Язык программирования: Python

Постановка задачи:

Пусть дана табличная функция y(x), представленная набором точек:
$\{(x\_i,\ y\_i) \mid i\ =\ 1,\ 2,\ \ldots,\ n\}$,
где:
• x_i — значения аргумента на отрезке [xmin; xmax],
• y_i — соответствующие значения функции,
• n — количество заданных точек.
Задача заключается в нахождении аппроксимирующей функции f(x), которая наилучшим образом описывает заданную табличную функцию y(x) на отрезке [xmin; xmax].
Требуется:
Выполнить аппроксимацию заданной табличной функции y(x) методом Эйткена.
Вычислить значения аппроксимирующей функции f(x) для набора точек x на отрезке [xmin; xmax].
Визуализировать решение, построив график, содержащий:
• исходные точки (x_i, y_i),
• график аппроксимирующей функции f(x).
Результатом должны быть:
• Аналитическое представление аппроксимирующей функции f(x).
• Таблица значений f(x) для заданного набора точек x.
• График, отображающий исходные данные и аппроксимирующую функцию.

Метод решения:

Метод Эйткена является эффективным способом интерполяции табличных функций. Основная идея метода заключается в последовательном уточнении значений интерполяционного многочлена с использованием рекуррентной формулы.

Значения интерполяционного полинома Лагранжа можно определить численно, без нахождения самого многочлена.

Для нахождения алгоритма таких вычислений, рассмотрим выражение

$$L_{0,1}(x) = \frac{\begin{vmatrix} y_0 & x_0 - x \\ y_1 & x_1 - x \end{vmatrix}}{x_1 - x_0}.$$

Оно будет полиномом первой степени, обращающимся в y_0 и y_1 при $x = x_0$ и $x = x_1$.

Аналогично

$$L_{1,2}(x) = \frac{\begin{vmatrix} y_1 & x_1 - x \\ y_2 & x_2 - x \end{vmatrix}}{x_2 - x_1}$$

будет полиномом первой степени, обращающимся в y_1 и y_2 при $x = x_1$ и $x = x_2$.

Далее получим, что выражение

$$L_{0,1,2}(x) = \frac{\begin{vmatrix} L_{0,1}(x) & x_0 - x \\ L_{1,2}(x) & x_2 - x \end{vmatrix}}{x_2 - x_0}$$

будет полиномом второй степени, обращающимся в y_0, y_1 и y_2 при $x = x_0, x = x_1$ и $x = x_2$.

Аналогично

$$L_{1,2,3}(x) = \frac{\begin{vmatrix} L_{1,2}(x) & x_1 - x \\ L_{2,3}(x) & x_3 - x \end{vmatrix}}{x_3 - x_1}$$

будет полиномом второй степени, обращающимся в y_1, y_2 и y_3 при $x = x_1$, $x = x_2$ и $x = x_3$.

Продолжая процесс построения многочленов далее, найдем, что

$$L_{0,1,2,\ldots,n}(x) = \frac{\begin{vmatrix} L_{0,1,2,\ldots,n-1}(x) & x_0 - x \\ L_{1,2,3,\ldots,n}(x) & x_n - x \end{vmatrix}}{x_n - x_0}$$

будет полиномом n-й степени, обращающимся в y_0, y_1, …, y_n при $x = x_0$, $x = x_1$, …, $x = x_n$.

где k — номер итерации.

Алгоритм:

1. Подготовка данных:
   Имеем набор точек (x[i], y[i]), где i = 0, 1, ..., n-1
   Задана точка x, для которой нужно найти приближенное значение функции
2. Инициализация:
   Создаем таблицу P размером n x n
   Заполняем первый столбец таблицы P известными значениями функции: P[i,0] = y[i]
3. Итеративный процесс:
   Для j от 1 до n-1:
   Для i от 0 до n-j-1:
   Вычисляем P[i,j] по формуле:
   P[i,j] = ((x - x[i+j]) P[i,j-1] + (x[i] - x) P[i+1,j-1]) / (x[i] - x[i+j])

4. Получение результата:
   Искомое приближенное значение функции находится в ячейке $P[0, n-1]$

5. Визуализация (опционально):
   Строим график, отображающий исходные точки и полученную аппроксимирующую функцию

Реализация программы:

Программа на Python реализует следующие функции:

- `aitken_interpolation(x, y, x_new)`:
  - ☐ Реализует алгоритм интерполяции Эйткена для одной точки.
  - ☐ Создает таблицу P и заполняет ее по формуле Эйткена.
  - ☐ Возвращает интерполированное значение для точки `x_new`.

- `approximate_function(x, y, x_new)`:
  - ☐ Применяет интерполяцию Эйткена ко всем точкам в x_new.

- Использует функцию `aitken_interpolation` для каждой точки.
- Возвращает массив аппроксимированных значений.

- `read_input_data(filename)`:
  - Читает входные данные из файла.
  - Извлекает значения x, y и границы интервала (x_min, x_max).
  - Возвращает эти данные в виде массивов numpy.
- `write_output_data(filename, x, y, x_new, y_new)`:
  - Записывает результаты в выходной файл.
  - Сохраняет исходные данные и аппроксимированные значения.
- `plot_approximation(x, y, x_new, y_new, x_min, x_max, filename)`:
  - Создает график с исходными точками и аппроксимированной функцией.
  - Отмечает границы интервала аппроксимации.
  - Сохраняет график в файл.
- `main()`:
  - Координирует работу всей программы.
  - Вызывает функции для чтения данных, аппроксимации, записи результатов и построения графика.

Пример кода:

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

def aitken_interpolation(x, y, x_new):
    n = len(x)
    P = np.zeros((n, n))
    P[:, 0] = y

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

```
            P[i, j] = ((x_new - x[i+j]) * P[i, j-1] +
(x[i] - x_new) * P[i+1, j-1]) / (x[i] - x[i+j])

    return P[0, n-1]

def approximate_function(x, y, x_new):
    return np.array([aitken_interpolation(x, y, xi) for
xi in x_new])

def read_input_data(filename):
    with open(filename, 'r') as f:
        lines = f.readlines()
        x = np.array([float(val) for val in
lines[0].split()])
        y = np.array([float(val) for val in
lines[1].split()])
        x_min, x_max = map(float, lines[2].split())
    return x, y, x_min, x_max

def write_output_data(filename, x, y, x_new, y_new):
    with open(filename, 'w', encoding='utf-8') as f:
        f.write("Исходные данные:\n")
        f.write(f"x: {x}\n")
        f.write(f"y: {y}\n\n")
        f.write("Аппроксимированные значения:\n")
        for xi, yi in zip(x_new, y_new):
            f.write(f"x: {xi:.2f}, y: {yi:.2f}\n")

def plot_approximation(x, y, x_new, y_new, x_min, x_max,
filename):
    plt.figure(figsize=(10, 6))
    plt.plot(x, y, 'ro', label='Исходные точки')
    plt.plot(x_new, y_new, 'b-', label='Аппроксимация по
Эйткену')
```

```
    plt.axvline(x=x_min, color='g', linestyle='-.',
label='Границы аппроксимации')
    plt.axvline(x=x_max, color='g', linestyle='-.')
    plt.xlabel('x')
    plt.ylabel('y')
    plt.title('Аппроксимация табличной функции методом
Эйткена')
    plt.legend()
    plt.grid(True)
    plt.savefig(filename)
    plt.close()

def main():
    x, y, x_min, x_max = read_input_data('input.txt')

    x_new = np.linspace(x_min, x_max, 100)

    y_new = approximate_function(x, y, x_new)

    write_output_data('output.txt', x, y, x_new, y_new)

    plot_approximation(x, y, x_new, y_new, x_min, x_max,
'approximation.png')

    print("Расчеты выполнены. Результаты сохранены в
файле output.txt, график сохранен в файле
approximation.png")

if __name__ == "__main__":
    main()
```

Тестовые данные:

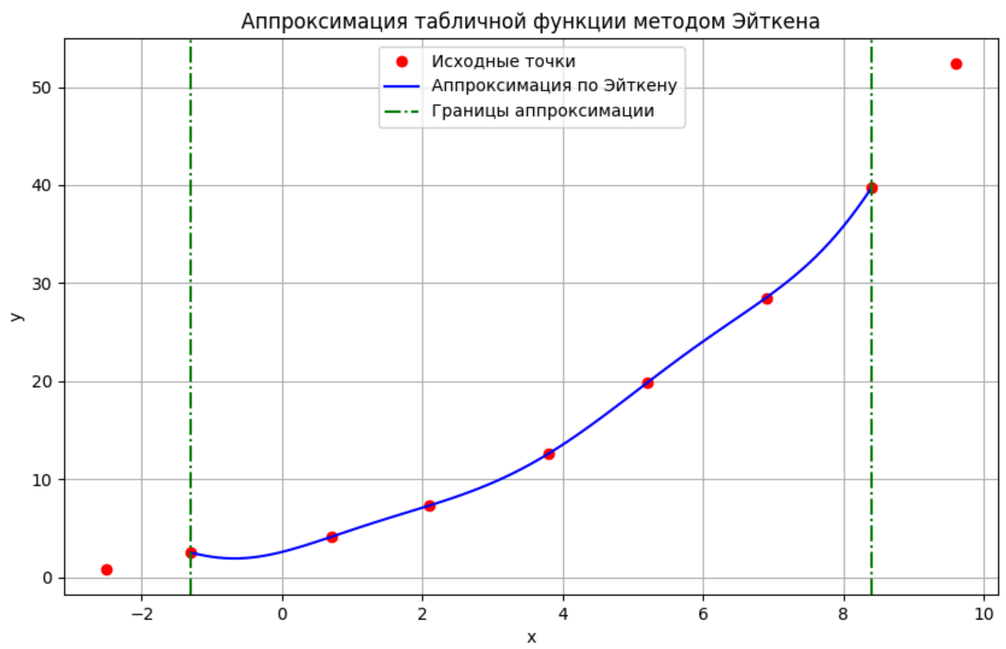
Для проверки работы программы используются следующие тестовые данные:

x	-2.5	-1.3	0.7	2.1	3.8	5.2	6.8	8.4	9.6
y	0.8	2.5	4.1	7.3	12.6	19.8	28.5	39.7	52.4

Границы	-1.3	8.4

Результат

Программа успешно строит аппроксимацию и выводит следующий результат:



Заключение:
Метод Эйткена является эффективным способом интерполяции табличных функций. Данный метод легко реализуется программно и позволяет получать приближенные значения функции в произвольных точках интервала. Важным преимуществом метода Эйткена является его

способность работать без явного вычисления коэффициентов интерполяционного многочлена.