МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ РОССИИ

ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ
УЧРЕЖДЕНИЕ
ВЫСШЕГО ПРОФЕССИОНАЛЬНОГО ОБРАЗОВАНИЯ

«САНКТ-ПЕТЕРБУРГСКИЙ ПОЛИТЕХНИЧЕСКИЙ УНИВЕРСИТЕТ ПЕТРА ВЕЛИКОГО»

Институт Компьютерных наук и Кибербезопасности

Направление 02.03.01 Математика и компьютерные науки

Вычислительная математика

Лабораторная работа №1

«Интерполяция»

Вариант №46

Студент группы
№5130201/30002: __________ Смирнов С.Р.

Преподаватель: __________ Пак В.Г.

«____»__________ 20___г.

Санкт-Петербург 2024

Содержание

Введение

Интерполяция — в вычислительной математике способ нахождения промежуточных значений величины по имеющемуся дискретному набору известных значений. Часто из наборов значений или данных требуется построить функцию, на которую могли бы с высокой точностью попадать другие получаемые значения.

1 Постановка задач

Цель: научиться вычислять приближённо табличную функцию с помощью интерполяционных многочленов

1. Для данной табличной функции вычислить приближённое значение в точке интерполяции с помощью многочлена Лагранжа и схемы Эйткена.

2. Для табличной функции из задания 1 вычислить приближённые значения в точках интерполяции с помощью подходящего многочлена Ньютона с разделёнными разностями.

3. Для данной табличной функции вычислить приближённые значения в точках интерполяции с помощью подходящего многочлена Ньютона с конечными разностями (точки x(1), x(2)) и наиболее подходящего из многочленов Гаусса, Стирлинга или Бесселя (точка x).

2 Задание 1

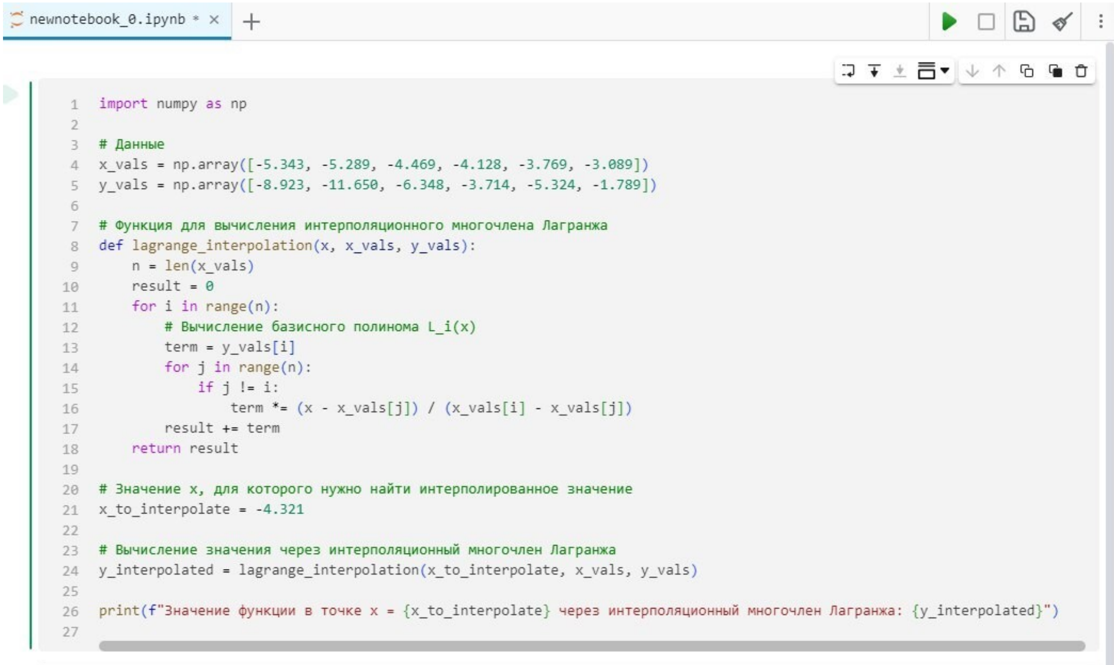
Даны несколько точек x некоторой функции и сооьветствующие им значения y. Можно увидеть эти значения в приведенной ниже таблице 1.

i	x	y
0	-5.343	-8.923
1	-5.289	-11.650
2	-4.469	-6.348
3	-4.128	-3.714
4	-3.769	-5.324
5	-3.089	-1.789

Таблица 1: Таблица значений функции в различных точках.

2.1 Нахождение значения функции с помощью интерполяционного многочлена Лагранжа

В первой части задания нужно вычислить приближённое значение в точке интерполяции с помощью многочлена Лагранжа. Сам многочлен Лагранжа выглядит так: $L_n(x) = \sum_{i=0}^{n} y_i \cdot l_{n,i}(x)$, где $l_{n,i}(x) = \prod_{\substack{j=0 \\ j\neq i}} \frac{x-x_j}{x_i-x_j}$. По известным нам точкам из таблицы, найдем значение функции в точке $x = -4,321$. Искать будем с помощью инструмента Engee. Ниже, на рисунке 1 , приведен код, который был написан для нахождения функции в точке $x = -4,321$.



```python
import numpy as np

# Данные
x_vals = np.array([-5.343, -5.289, -4.469, -4.128, -3.769, -3.089])
y_vals = np.array([-8.923, -11.650, -6.348, -3.714, -5.324, -1.789])

# Функция для вычисления интерполяционного многочлена Лагранжа
def lagrange_interpolation(x, x_vals, y_vals):
    n = len(x_vals)
    result = 0
    for i in range(n):
        # Вычисление базисного полинома L_i(x)
        term = y_vals[i]
        for j in range(n):
            if j != i:
                term *= (x - x_vals[j]) / (x_vals[i] - x_vals[j])
        result += term
    return result

# Значение x, для которого нужно найти интерполированное значение
x_to_interpolate = -4.321

# Вычисление значения через интерполяционный многочлен Лагранжа
y_interpolated = lagrange_interpolation(x_to_interpolate, x_vals, y_vals)

print(f"Значение функции в точке x = {x_to_interpolate} через интерполяционный многочлен Лагранжа: {y_interpolated}")
```

Рис. 1: Вычисление значения функции с помощью многочлена Лагранжа

Этот код определяет массивы значений x и y; реализует функцию для вычисления интерполяционного многочлена Лагранжа и вычисляет это значения в точке $x = -4,321$. Округлим полученное значение до 3-х знаков после запятой и получим, что $y(-4,321) = -4,584$

2.2 Нахождение значения функции с помощью схемы Эйткена

Вычислительная схема Эйткена предназначена для пошагового рекуррентного вычисления значения интерполяционного многочлена Лагранжа L_n в точке x. Для вычисления значения функции с использованием метода Эйткена нужно использовать формулу Эйткена, которая выглядит так:

$$P(x) = \frac{(x - x_0) \cdot L_1(x) - (x - x_1) \cdot L_0(x)}{(x - x_0) - (x - x_1)}$$

где:

- $L_0(x)$ и $L_1(x)$ — это интерполяционные многочлены Лагранжа, построенные для каждой пары точек, и
- (x_0, y_0) и (x_1, y_1) — соответствующие точки, для которых рассчитываются эти многочлены.

Значение функции в точке $x = -4,321$ было вычислено в Engee посредством кода, указанного на рисунке 2.

```python
# Данные
x_vals = np.array([-5.343, -5.289, -4.469, -4.128, -3.769, -3.089])
y_vals = np.array([-8.923, -11.650, -6.348, -3.714, -5.324, -1.789])

# Функция для применения схемы Эйткена
def aitken_interpolation(x, x_vals, y_vals):
    n = len(x_vals)

    # Создаем таблицу для хранения промежуточных результатов
    A = np.zeros((n, n))

    # Заполняем первую колонку таблицы значениями y
    for i in range(n):
        A[i][0] = y_vals[i]

    # Для каждой колонки заполняем значения по схеме Эйткена
    for j in range(1, n):
        for i in range(n - j):
            A[i][j] = ((x - x_vals[i]) * A[i+1][j-1] - (x - x_vals[i+j]) * A[i][j-1]) / (x_vals[i+j] - x_vals[i])

    # Возвращаем верхний элемент последней строки (это и есть искомое значение)
    return A[0][n-1]

# Значение x, для которого нужно найти интерполированное значение
x_to_interpolate = -4.321

# Вычисление значения через схему Эйткена
y_aitken = aitken_interpolation(x_to_interpolate, x_vals, y_vals)

print(f"Значение функции в точке x = {x_to_interpolate} через схему Эйткена: {y_aitken}")
```

Рис. 2: Вычисление значения функции с помощью схемы Эйткена

Функция aitken_interpolation реализует схему Эйткена. Вначале строится таблица, где для каждого значения y_iна первом шаге заполняются начальные значения, а затем происходит вычисление значений с помощью формулы Эйткена. В конечном счете было получено то же самое значение, что и с помощью интерполяционного многочлена Лагранжа: $y(-4,321) = -4,584$

3 Задание 2

В этом задании нужно по таблице 1 через интерполяционные многочлены Ньютона с разделенными разностями вычислить значения в точках $x^{(1)} = -5,282$ и $x^{(2)} = -3,002$.

3.1 Вычисление значения функции с помощью первого многочлена Ньютона

Для нахождения значения в точке $x^{(1)} = -5,282$ будет использован интерполяционный многочлен Ньютона с разделенными разностями первого вида. По другому он называется многочленом для интерполирования вперед и имеет вид:

$$P_1(x) = f(x_0) + f(x_0, x_1)(x - x_0) + f(x_0, x_1, x_2)(x - x_0)(x - x_1) + ... + f(x_0, ..., x_n)(x - x_0)...(x - x_{n-1})$$

Используем его, так как точка $x^{(1)} = -5,282$ находится ближе к началу таблицы. Для нахождения значения используем среду Engee. На рисунке 3 приведен код, с помощью которого было посчитано значение в точке $x^{(1)} = -5,282$.

```
import numpy as np
x_values = np.array([-5.343, -5.289, -4.469, -4.128, -3.769, -3.089])
y_values = np.array([-8.923, -11.650, -6.348, -3.714, -5.324, -1.789])
# Функция для вычисления разделенных разностей
def divided_difference(x, y):
    n = len(x)
    diff_table = np.zeros((n, n))
    diff_table[:, 0] = y  # Первая колонка это y-значения

    # Заполнение таблицы разделенных разностей
    for j in range(1, n):
        for i in range(n - j):
            diff_table[i, j] = (diff_table[i + 1, j - 1] - diff_table[i, j - 1]) / (x[i + j] - x[i])

    return diff_table
# Функция для вычисления значения с использованием первого многочлена Ньютона
def newton_interpolation(x_values, y_values, x):
    # Получаем таблицу разделенных разностей
    diff_table = divided_difference(x_values, y_values)

    # Начинаем с первого значения (f(x0))
    result = y_values[0]
    term = 1
    for i in range(1, len(x_values)):
        term *= (x - x_values[i - 1])
        result += term * diff_table[0, i]

    return result
x_point = -5.282
# Вычисление значения функции в точке x = -5.282
result = newton_interpolation(x_values, y_values, x_point)
result
```

Рис. 3: Вычисление значения функции с помощью первого многочлена Ньютона

Функция divided_difference строит таблицу разделенных разностей для набора значений x и y.newton_interpolation — функция для вычисления значения интерполяционного многочлена Ньютона в заданной точке x, используя таблицу разделенных разностей. В конечном счете было получено значение $y(-5,282) = -11,944$.

3.2 Вычисление значения функции с помощью второго многочлена Ньютона

Для нахождения значения в точке $x^{(2)} = -3,002$ будет использован интерполяционный многочлен Ньютона с разделенными разностями второго вида. По другому он называется многочленом для интерполирования назад и имеет вид:

$$P_2(x) = f(x_n)+f(x_{n-1},x_n)(x-x_n)+f(x_{n-2},x_{n-1},x_n)(x-x_n)(x-x_{n-1})+...+f(x_0,...,x_n)(x-x_n)...(x-$$

Используем его, так как точка $x^{(2)} = -3,002$ находится ближе к концу таблицы. Для нахождения значения используем среду Engee. На рисунке 4 приведен код, с помощью которого было посчитано значение в точке $x^{(2)} = -3,002$.

```
import numpy as np
x_values = np.array([-5.343, -5.289, -4.469, -4.128, -3.769, -3.089])
y_values = np.array([-8.923, -11.650, -6.348, -3.714, -5.324, -1.789])
# Функция для вычисления разделенных разностей
def divided_difference(x, y):
    n = len(x)
    diff_table = np.zeros((n, n))
    diff_table[:, 0] = y

    # Заполнение таблицы разделенных разностей
    for j in range(1, n):
        for i in range(n - j):
            diff_table[i, j] = (diff_table[i + 1, j - 1] - diff_table[i, j - 1]) / (x[i + j] - x[i])

    return diff_table
# Функция для вычисления значения с использованием многочлена Ньютона
def newton_interpolation(x_values, y_values, x):
    # Получаем таблицу разделенных разностей
    diff_table = divided_difference(x_values, y_values)
    result = y_values[0]
    term = 1  # Это множитель для каждого члена

    # Добавляем каждый следующий член многочлена
    for i in range(1, len(x_values)):
        term *= (x - x_values[i - 1])
        result += term * diff_table[0, i]
    return result
x_point = -3.002
# Вычисление значения функции в точке x = -3.002 с использованием многочлена Ньютона
result = newton_interpolation(x_values, y_values, x_point)
result
```

Рис. 4: Вычисление значения функции с помощью второго многочлена Ньютона

Функция divided_difference строит таблицу разделенных разностей для набора значений x и y.newton_interpolation — функция для вычисления значения интерполяционного многочлена Ньютона в заданной точке x, используя таблицу разделенных разностей. В конечном счете было получено значение $y(-3,002) = 1,047$.

4 Задание 3

В этом задании нужно по фукнции $y = arcsin(2x)$ ограниченной от -0,5 до 0,5 с шагом h=0,2 через интерполяционные многочлены Ньютона с конечными разностями вычислить значения в точках $x^{(1)} = -0,28$ и $x^{(2)} = 0,28$. Также нужно вычислить значение функции в точке $x = -0,02$ через одтн из многочленов Гаусса, Стирлинга или Бесселя. По таблице 2 можно заметить, что число узлов в ней четное, следовательно, стоит использовать многочлен Бесселя.

i	x	y
-2	-0,5	-1,57
-1	-0,3	-0,64
0	-0,1	-0,20
1	0,1	0,20
2	0,3	0,64
3	0,5	1,57

Таблица 2: Таблица значений функции $y = arcsin(2x)$ в различных точках.

4.1 Вычисление значения функции с помощью первого многочлена Ньютона

Найдем значения в точке $x^{(1)} = -0,28$ с помощью первого многочлена Ньютона с конечными разностями. Для этого воспользуемся средой Engee. Код приведен на рисунке 5.

```python
import numpy as np
x_values = np.array([-0.5, -0.3, -0.1, 0.1, 0.3, 0.5])
y_values = np.array([-1.57, -0.64, -0.20, 0.20, 0.64, -1.57])

# Функция для вычисления конечных разностей
def finite_differences(x, y):
    n = len(x)
    diff_table = np.zeros((n, n))
    diff_table[:, 0] = y
    # Заполнение таблицы конечных разностей
    for j in range(1, n):
        for i in range(n - j):
            diff_table[i, j] = diff_table[i + 1, j - 1] - diff_table[i, j - 1]

    return diff_table

# Функция для вычисления значения интерполяции с использованием многочлена Ньютона с конечными разностями
def newton_interpolation(x_values, y_values, x):
    diff_table = finite_differences(x_values, y_values)
    # Начинаем с первого значения f(x0)
    result = y_values[0]
    term = 1
    n = len(x_values)

    # Добавляем каждый следующий член многочлена, учитывая факториал
    for i in range(1, n):  # Теперь проходим по всем точкам
        term *= (x - x_values[i - 1])  # Множитель для текущего члена
        # Для второго и более порядков разностей делим на факториал
        result += term * diff_table[0, i] / np.math.factorial(i)

    return result
x_point = -0.28
result = newton_interpolation(x_values, y_values, x_point)
result
```

Рис. 5: Вычисление значения функции с помощью первого многочлена Ньютона с конечными разностями

В конечном счете было получено значение $y(-0,28) = -0,59$.

4.2 Вычисление значения функции с помощью второго многочлена Ньютона

Найдем значения в точке $x^{(2)} = 0,28$ с помощью второго многочлена Ньютона с конечными разностями. Для этого воспользуемся средой Engee. Код приведен на рисунке 6.

```python
import numpy as np
x_values = np.array([-0.5, -0.3, -0.1, 0.1, 0.3, 0.5])
y_values = np.array([-1.57, -0.64, -0.20, 0.20, 0.64, -1.57])

# Функция для вычисления конечных разностей
def finite_differences(x, y):
    n = len(x)
    diff_table = np.zeros((n, n))
    diff_table[:, 0] = y
    # Заполнение таблицы конечных разностей
    for j in range(1, n):
        for i in range(n - j):
            diff_table[i, j] = diff_table[i + 1, j - 1] - diff_table[i, j - 1]

    return diff_table

# Функция для вычисления значения интерполяции с использованием многочлена Ньютона с конечными разностями
def newton_interpolation(x_values, y_values, x):
    diff_table = finite_differences(x_values, y_values)
    # Начинаем с первого значения f(x0)
    result = y_values[0]
    term = 1
    n = len(x_values)

    # Добавляем каждый следующий член многочлена, учитывая факториал
    for i in range(1, n):  # Теперь проходим по всем точкам
        term *= (x - x_values[i - 1])  # Множитель для текущего члена
        # Для второго и более порядков разностей делим на факториал
        result += term * diff_table[0, i] / np.math.factorial(i)

    return result
x_point = 0.28
result = newton_interpolation(x_values, y_values, x_point)
result
```

Рис. 6: Вычисление значения функции с помощью второго многочлена Ньютона с конечными разностями

В конечном счете было получено значение $y(0,28) = 0,59$.

4.3 Вычисление значения функции с помощью многочлена Бесселя

В данном случае будет удобно использовать многочлен Бесселя, так как число узлов в таблице четное. Он имеет следующий вид:

$$P_n = y_0 + \frac{\delta y_0}{1!}q + \frac{\delta^2 y_{-1}}{2!}q(q-1) + \frac{\delta^3 y_{-1}}{3!}q(q-1)(q+1) + \frac{\delta^4 y_{-2}}{3!}q(q-1)(q+1)(q-2)$$

, где $q = \frac{x-a}{h}$ При x=-0,02 q будет равно 0,4. Используем среду Enge, чтобы вычислить значение $x = -0,02$. Код приведен на рисунке 7.

```python
import math

# Данные
y_values = np.array([-1.57, -0.64, -0.20, 0.20, 0.64, -1.57])

# Функция для вычисления конечных разностей
def compute_deltas(y):
    n = len(y)
    deltas = []
    for i in range(1, n):
        delta = [y[j + 1] - y[j] for j in range(n - i)]
        deltas.append(delta)
        y = delta  # Обновляем для вычисления разностей более высоких порядков
    return deltas

# Рассчитаем все разности
deltas = compute_deltas(y_values)

# Задаем q = 0.4
q = 0.4
# Вычисляем значение многочлена Бесселя
def bessel_polynomial(deltas, q):
    P_n = y_values[0]  # Начинаем с y_0
    term = 1  # Начинаем с первого множителя
    # Первая разность (для y_0)
    P_n += deltas[0][0] * q

    # Добавляем остальные члены многочлена
    for i in range(1, len(deltas)):
        term *= (q - i + 1)
        P_n += (deltas[i][0] / math.factorial(i + 1)) * term
    return P_n
# Вычисление значения многочлена Бесселя
result = bessel_polynomial(deltas, q)
print(f"Значение интерполяционного многочлена в точке q={q}: {result}")
```

Рис. 7: Вычисление значения функции с помощью многочлена Бесселя

В конечном счете было получено значение $y(-0,02) = -0,04$.

Заключение

В ходе выполнения лабораторной работы были изучены различные способы нахождения значений функции с помощью интерполяционных многочленов Лагранжа, Ньютона, Гаусса, Стирлинга, Бесселя. Также была освоена среда Engee и jupyter notebook. Эти навыки могут быть использованы ы профессиональной деятельности.