МИНИСТЕРСТВО НАУКИ И ВЫСШЕГО ОБРАЗОВАНИЯ
РОССИЙСКОЙ ФЕДЕРАЦИИ

Федеральное государственное автономное образовательное учреждение высшего 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 36

Студент,
группы 5130201/30002 ________________ Михайлова А. А.

Руководитель, ________________ Пак В. Г.

«_____»________________ 20__ г.

Санкт-Петербург, 2024, осенний семестр

1 Задание 1

Для данной табличной функции вычислить приближённое значение в точке интерполяции с помощью многочлена Лагранжа, а затем с помощью схемы Эйткена.

Точка интерполяции x = 12,721.

x_i	11,153	11,454	11,673	11,879	12,009	12,231	12,549
y_i	−3,234	5,321	−1,123	0,393	8,939	141,231	15,001

Рис. 1: Табличная функция

1.1 Вычисление с помощью многочлена Лагранжа

```julia
function lagrange_interpolation(x_points, y_points, x)
    n = length(x_points)
    result = 0.0
    for i in 1:n
        term = y_points[i]
        for j in 1:n
            if i != j
                term *= (x - x_points[j]) / (x_points[i] - x_points[j])
            end
        end
        result += term
    end
    return result
end

x_points = [11.153, 11.454, 11.673, 11.879, 12.009, 12.231, 12.549]
y_points = [-3.234, 5.321, -1.123, 0.393, 8.939, 141.231, 15.001]
x_interpolation = 12.721

lagrange_value = lagrange_interpolation(x_points, y_points, x_interpolation)

println("Приближенное значение (многочлен Лагранжа): ", lagrange_value)

```

1.1.1 Ответ

```
Приближенное значение (многочлен Лагранжа): -2109.462707520738
```

1.2 Вычисление с помощью схемы Эйткена

```julia
function aitken_interpolation(x_points, y_points, x)
    n = length(x_points)
    P = zeros(n, n)

    for i in 1:n
        P[i, 1] = y_points[i]
    end

    for j in 2:n
        for i in 1:(n-j+1)
            P[i, j] = ( (x - x_points[i+j-1]) * P[i, j-1] - (x - x_points[i]) * P[i+1, j-1] ) / (x_points[i] - x_points[i+j-1])
        end
    end

    return P[1, n]
end

x_points = [11.153, 11.454, 11.673, 11.879, 12.009, 12.231, 12.549]
y_points = [-3.234, 5.321, -1.123, 0.393, 8.939, 141.231, 15.001]
x_interpolation = 12.721

aitken_value = aitken_interpolation(x_points, y_points, x_interpolation)

println("Приближенное значение (схема Эйткена): ", aitken_value)
```

1.2.1 Ответ

```
Приближенное значение (схема Эйткена): -2109.462707520738
```

2 Задание 2

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

Многочлен Ньютона с разделёнными разностями $x^{(1)} = 0,66$, $x^{(2)} = 0,66$.

```python
import numpy as np
x_values = np.array([11.153, 11.454, 11.673, 11.879, 12.009, 12.231, 12.549])
y_values = np.array([-3.234, 5.321, -1.123, 0.393, 8.939, 141.231, 15.001])
def divided_differences(x, y):
    n = len(y)
    coeffs = np.zeros((n, n))
    coeffs[:, 0] = y
    for j in range(1, n):
        for i in range(n - j):
            coeffs[i][j] = (coeffs[i + 1][j - 1] - coeffs[i][j - 1]) / (x[i + j] - x[i])
    return coeffs[0]
def newton_interpolation(x_values, y_values, x):
    coeffs = divided_differences(x_values, y_values)
    n = len(coeffs)
    result = 0
    for i in range(n):
        term = coeffs[i]
        for j in range(i):
            term *= (x - x_values[j])
        result += term
    return result
```

```python
x1 = 12.520
x2 = 11.512
approx_value_x1 = newton_interpolation(x_values, y_values, x1)
approx_value_x2 = newton_interpolation(x_values, y_values, x2)
print(f'Приближенное значение в точке x1: {approx_value_x1}')
print(f'Приближенное значение в точке x2: {approx_value_x2}')
```

2.1 Ответ

```
Приближенное значение в точке x1: 133.68253280445788
Приближенное значение в точке x2: -2.257643189844589
```

3 Задание 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).

$x^{(1)} = 0,79$, $x^{(2)} = 4,80$, $x = 3,22$

36.	$e - (\ln(x))^2$	1	1	5

```python
import numpy as np
def f(x):
    return np.exp(1) - (np.log(x))**2
a = 1
b = 5
h = 1
x1 = 0.79
x2 = 4.80
x = 3.22
x_values = np.arange(a, b + h, h)
y_values = f(x_values)
def finite_differences(y):
    n = len(y)
    diff_table = np.zeros((n, n))
    diff_table[:, 0] = y

    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
diff_table = finite_differences(y_values)
```

```python
def newton_interpolation(x, x_values, diff_table):
    n = len(x_values)
    result = diff_table[0, 0]
    product_term = 1

    for i in range(1, n):
        product_term *= (x - x_values[i - 1])
        result += (diff_table[0, i] / np.math.factorial(i)) * product_term

    return result
approx_x1 = newton_interpolation(x1, x_values, diff_table)
approx_x2 = newton_interpolation(x2, x_values, diff_table)
print(f"Приближенное значение в точке x1 = {x1}: {approx_x1:.2f}")
print(f"Приближенное значение в точке x2 = {x2}: {approx_x2:.2f}")
def gauss_interpolation(x, x_values, y_values):
    idx = np.searchsorted(x_values, x)
    if idx == 0:
        return y_values[0]
    elif idx == len(y_values):
        return y_values[-1]
    else:
        x0, x1 = x_values[idx - 1], x_values[idx]
        y0, y1 = y_values[idx - 1], y_values[idx]
```

```
        return y0 + (y1 - y0) * (x - x0) / (x1 - x0)
gauss_approx = gauss_interpolation(x, x_values, y_values)
print(f"Приближенное значение в точке x = {x}: {gauss_approx:.2f}")
```

3.1 Ответ

```
Приближенное значение в точке x1 = 0.79: 2.75
Приближенное значение в точке x2 = 4.8: 0.26
Приближенное значение в точке x = 3.22: 1.35
```