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

Высшая школа технологий искусственного интеллекта

Направление: 02.03.01 Математика и компьютерные науки

Отчет о выполнении лабораторной работы №1
«Интерполяция»
Дисциплина «Вычислительная математика»
Вариант 14.

Выполнил студент группы
№5130201/30001 _____ Мелещенко С.И.

Проверил _____ Пак В. Г.

Санкт-Петербург, 2024

Введение

Данный отчет содержит в себе описание выполнения лабораторной работы №1 «Интерполяция» по дисциплине «Вычислительная математика».

Интерполяция решает задачу нахождения значения функции в точке, для которой нет известного значения, на основе её значений в других точках. Она часто применяется для обработки данных, создания численных моделей и предсказания значений между измеренными точками. К числу основных задач интерполяции относят восстановление функции по конечному набору данных и вычисление приближенных значений функции.

Интерполяционный полином Лагранжа строится для точного прохождения через заданные точки. Он представляет собой сумму, где каждая точка включает коэффициент, зависящий от всех остальных точек.

Схема Эйткена — это численный метод, позволяющий последовательно вычислять приближенное значение функции в новой точке с использованием интерполяции Лагранжа. Она основывается на многоступенчатом процессе, в котором последовательно добавляются точки.

Полином Ньютона с разделёнными разностями строится для упрощения расчётов в полиномиальной интерполяции, позволяя добавлять новые точки без пересчёта всего многочлена. Разделённые разности формируют таблицу, на основе которой можно составить интерполяционный полином.

Полином Ньютона с конечными разностями используется для равномерно распределённых точек. Здесь вычисляют конечные разности, которые упрощают расчет коэффициентов. Полином Ньютона можно составлять "с конца"или "с начала что зависит от положения интерполируемой точки относительно известного набора данных.

Полиномы Гаусса, Стирлинга и Бесселя применяются для интерполяции в точке, близкой к середине интервала данных. В этих методах используются конечные разности, что позволяет повысить точность на центральных участках.

Цель: научиться вычислять приближённо табличную функцию с помощью интерполяционных многочленов

Задание:

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).

1 Решение

1.1 Задание 1

x_i	−1,93	−1,46	−0,87	−0,53	−0,29	−0,05
y_i	18,3456	19,9472	21,91492	21,8492	23,1236	25,8484

x = -0.15

Для данной табличной функции вычислить приближённое значение в точке интерполяции с помощью многочлена Лагранжа и схемы Эйткена. Для вычисления была использована среда Engee.

Листинг 1: Лагранж

```
x = [−1.93, −1.46, −0.87, −0.53, −0.29, −0.05]
y = [18.3456, 19.9472, 21.91492, 21.8492, 23.1236, 25.8484]
x_interp = −0.15

function lagrange(x, y, x_interp)
    n = length(x)
    result = 0.0
    for i in 1:n
        term = y[i]
        for j in 1:n
            if j != i
                term *= (x_interp − x[j]) / (x[i] − x[j])
            end
        end
        result += term
    end
    return result
end
lagrange_result = lagrange(x, y, x_interp)
println("$lagrange_result")
```

Результат: 24.57.

Листинг 2: Эйткин

```
x = [−1.93, −1.46, −0.87, −0.53, −0.29, −0.05]
y = [18.3456, 19.9472, 21.91492, 21.8492, 23.1236, 25.8484]
x_interp = −0.15

function aitken(x, y, x_interp)
    n = length(x)
    p = copy(y)
    for i in 1:n−1
        for j in 1:n−i
            p[j] = ((x_interp − x[j+i])\
            * p[j] + (x[j] − x_interp) * p[j+1]) / (x[j] − x[j+i])
        end
    end
    return p[1]
end
```

```
aitken_result = aitken(x, y, x_interp)
println("$aitken_result")
```

Результат: 24.57.

1.2 Задание 2

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

x(1) = -1.77

x(2) = -0.15

Для x(1) логично использовать первый многочлен Ньютона, для x(2) - второй.

Листинг 3: Ньютон с разделенными разностями

```
x_points1 = -1.77
x_points2 = -0.15
function divided_differences(x, y)
    n = length(x)
    dd_table = [y[i] for i in 1:n]
    for j in 2:n
        for i in n:-1:j
            dd_table[i] = (dd_table[i] - dd_table[i-1]) / (x[i] - x[i-j+1])
        end
    end
    return dd_table
end
function newton_interpolation_start(x, y, x_interp)
    dd = divided_differences(x, y)
    n = length(dd)
    result = dd[1]
    term = 1.0
    for i in 1:(n-1)
        term *= (x_interp - x[i])
        result += dd[i+1] * term
    end
    return result
end
function newton_interpolation(x, y, x_interp)
    dd = divided_differences(x, y)
    n = length(dd)
    result = dd[n]
    for i in n-1:-1:1
        result = result * (x_interp - x[i]) + dd[i]
    end
    return result
end
    result1 = newton_interpolation_start(x, y, x_interp1)
    result2 = newton_interpolation(x, y, x_interp2)
    println("$result1")
    println("$result2")
```

Результаты: 17.70, 24.57.

1.3 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).

Табличная функция f(x) = cosec(2x).

Шаг h = 0.1.

Интервал от a = 0.5 до b = 1.0.

Листинг 4: Ньютон с конечными разностями

```
x_values = [0.5, 0.6, 0.7, 0.8, 0.9, 1.0]
    y_values = [1.19, 1.07, 1.01, 1.00, 1.03, 1.10]

    function newton_forward_interpolation(x_values, y_values, x)
        n = length(x_values)
        F = zeros(n, n)
        F[:, 1] .= y_values
        for j in 2:n
            for i in 1:(n - j + 1)
                F[i, j] = F[i + 1, j - 1] - F[i, j - 1]
            end
        end
        index = findfirst(x -> x >= x, x_values)
        if index === nothing
            error("")
        end
        result = F[1, 1]
        term = 1.0
        for i in 1:(n - 1)
            term *= (x - x_values[i])
            result += F[1, i + 1] * term
        end

        return result
    end

function newton_backward_interpolation(x_values, y_values, x)
    n = length(x_values)
    F = zeros(n, n)
    F[:, 1] .= y_values

    for j in 2:n
        for i in 1:(n - j + 1)
            F[i, j] = F[i + 1, j - 1] - F[i, j - 1]
        end
    end
    index = findlast(x -> x <= x, x_values)
    if index === nothing
        error("")
    end
    result = F[index, 1]
    term = 1.0
    for i in 1:(index - 1)
        term *= (x - x_values[index - i])
        result += F[index - i, i + 1] * term / factorial(i)
    end
```

```
    return result
end
x_interp1 = 0.4
x_interp2 = 0.99

y_interp1 = newton_forward_interpolation(x_values, y_values, x_interp1)
y_interp2 = newton_backward_interpolation(x_values, y_values, x_interp2)

@printf( y_interp1)
@printf( y_interp2)
```

Результат: 1.21, 1.11.

Для точки x = 0.75 был выбран метод Гаусса, так как точка находится ровно по середине таблицы.

Листинг 5: Гаусс

```
function gauss_interpolation(x_vals, y_vals, x_interp)
    n = length(x_vals)
    diff_table = finite_differences(y_vals)
    h = x_vals[2] - x_vals[1]
    mid_index = div(n, 2)
    t = (x_interp - x_vals[mid_index]) / h
    interpolated_value = y_vals[mid_index]
    term = 1.0
    for k in 1:(n - 1)
        if k % 2 == 1
            idx = mid_index - div(k, 2)
        else
            idx = mid_index + div(k, 2)
        end
        term *= (t - k + 1) / k
        interpolated_value += term * diff_table[idx, k + 1]
    end
    return interpolated_value
end
approx_g = gauss_interpolation(x_vals, y_vals, x_interp)
@printf approx_g
```

Результат: 0.99.

2 Вывод

Были получены все результаты.

```
Значение в точке -0.15 с помощью многочлена Лагранжа: 24.57437862111489
Значение в точке -0.15 с помощью схемы Эйткена: 24.574378621114892
Значение в точке -1.77: 17.6961385589063
Значение в точке -0.15: 24.57437862111489
Приближенное значение в x = 0.40: y = 1.20844
Приближенное значение в x = 0.99: y = 1.10996
Приближенное значение в точке x = 0.75 с помощью многочлена Гаусса: 0.99219
```

Интерполяция позволяет находить значения функции в точках, где данные неизвестны, используя информацию из других точек. В ходе работы мы рассмотрели основные методы интерполяции: полином Лагранжа, схема Эйткена, полиномы Ньютона с разделёнными и конечными разностями, а также полиномы Гаусса, Стирлинга и Бесселя.

Цель работы заключалась в изучении методов интерполяции и их применении для вычисления приближённых значений функции. Задачи, поставленные в начале, включали:

1. Вычисление приближённых значений с помощью многочлена Лагранжа и схемы Эйткена.

2. Применение полинома Ньютона с разделёнными разностями для вычисления приближённых значений в точках интерполяции.

3. Применение многочлена Ньютона с конечными разностями и полиномов Гаусса, Стирлинга или Бесселя для более точной интерполяции.

В результате выполнения этих задач были получены приближённые значения функции в различных точках, что подтвердило эффективность каждого из методов интерполяции.