Федеральное государственное автономное образовательное учреждение высшего образования **«Национальный исследовательский университет ИТМО»**

Факультет Программной Инженерии и Компьютерной Техники

**Лабораторная работа по вычислительной математике №5**

Интерполяция функции

Преподаватель: Малышева Татьяна Алексеевна

Выполнила: Голованова Дарья Владимировна

Группа: Р3222

Санкт-Петербург,

2022г

Цель лабораторной работы:

Цель лабораторной работы: решить задачу интерполяции, найти значения функции при заданных значениях аргумента, отличных от узловых точек.

Для исследования использовать:

· многочлен Лагранжа;

· многочлен Ньютона;

· многочлен Гаусса.

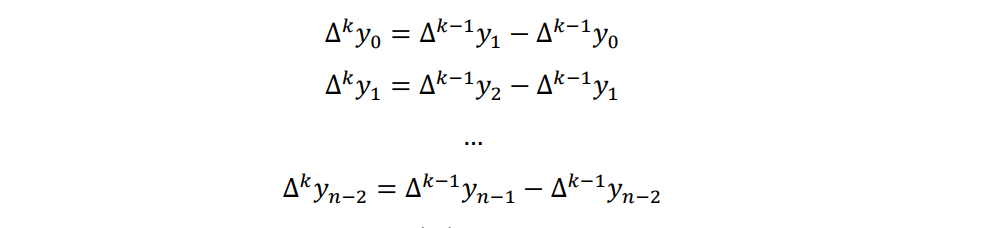
# Описание использованного метода:

Интерполяцией называют такую разновидность аппроксимации, при которой кривая построенной функции проходит через имеющиеся точки.

Многочлен Ньютона с конечными разностями:

Интерполирующий полином ищется в виде: 

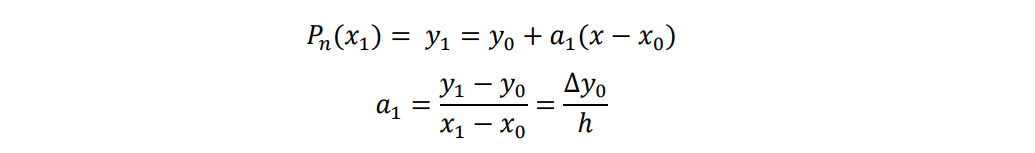
Построение многочлена сводится к определению коэффициентов 𝑎𝑖 При записи коэффициентов пользуются конечными разностями. Конечные разности первого порядка пишутся в виде:



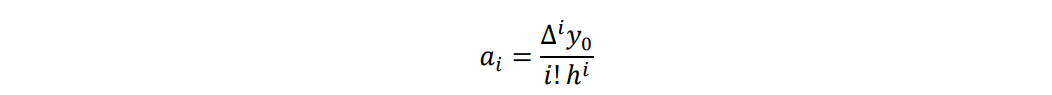
Коэффициенты 𝑎𝑛находятся из 𝑃𝑛 (𝑥𝑖 ) = 𝑦𝑖 . Находим 𝑎0полагая, что x = x0



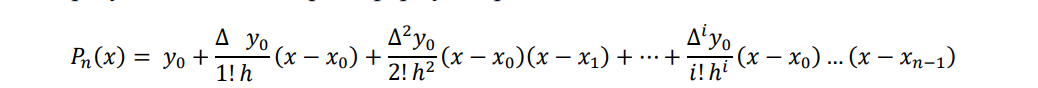
Далее подставляем значение x = 𝑥1, получим



Общая формула нахождения 𝑎𝑖:



В результате самая первая формула примет вид:

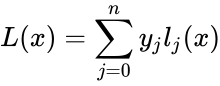


Данный многочлен называют первым полиномом Ньютона.

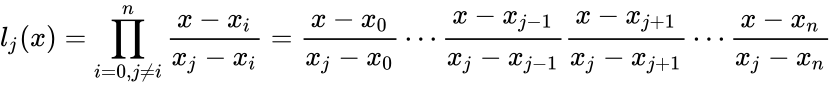
Многочлен Лагранжа:

Интерполяционный многочлен Лагранжа — многочлен минимальной степени, принимающий данные значения в данном наборе точек. Для n+1 пар чисел {\displaystyle (x_{0},y_{0}),(x_{1},y_{1})\dots (x_{n},y_{n})} , где все xi различны, существует единственный многочлен L(x) степени не более n, для которого {\displaystyle L(x_{i})=y_{i}}.

Лагранж предложил способ вычисления таких многочленов:



где базисные полиномы определяются по формуле:



Легко видеть, что {\displaystyle l_{j}(x)} обладают такими свойствами:

* Это полиномы степени n
* {\displaystyle l_{j}(x_{j})=1}
* {\displaystyle l_{j}(x_{i})=0} при {\displaystyle i\neq j}

Отсюда следует, что L(x), как [линейная комбинация](https://math.fandom.com/ru/wiki/%D0%9B%D0%B8%D0%BD%D0%B5%D0%B9%D0%BD%D0%B0%D1%8F_%D0%BA%D0%BE%D0%BC%D0%B1%D0%B8%D0%BD%D0%B0%D1%86%D0%B8%D1%8F) {\displaystyle l_{j}(x)}, может иметь степень не больше n, и {\displaystyle L(x_{j})=y_{j}}

# Вычислительная реализация:

# Листинг численного метода:

def newton(x: list, y: list, x0: float):

    if not check\_nodes(x):

        raise Exception('Узлы не являются равноотстоящими, метод Ньютона с конечными разностями не применим.')

    if x0 in x:

        return y[x.index(x0)], ""

    dy = get\_finite\_differences(id(y))

    h = (x[1] - x[0])

    nearest\_point = -1

    for index in range(len(x)):

        if x[index] >= x0:

            nearest\_point = index - 1

            break

    result = 0

    if x0 - x[0] < x[-1] - x0:

        t = (x0 - x[nearest\_point]) / h

        for i in range(len(dy) - nearest\_point):

            result += dy[nearest\_point][i] \* get\_t(i, t)

    else:

        try:

            nearest\_point += 1

            t = (x0 - x[nearest\_point]) / h

            for i in range(nearest\_point, -1, -1):

                result += dy[i][nearest\_point - i] \* get\_t(nearest\_point - i, t, back=True)

        except Exception:

            return -10, ""

    return result, ""

def lagrange(x: list, y: list, x0: float):

    result = 0

    for j in range(len(y)):

        mul = 1

        for i in range(len(x)):

            mul \*= (x0 - x[i]) / (x[j] - x[i]) if i != j else 1

        result += y[j] \* mul

    return result, ""

# Функция для вычисления значения с использованием

# Формула Стирлинга (нечет)

def stirling(x, fx, x1):

    message = ""

    d = 1

    temp1 = 1

    temp2 = 1

    k = 1

    n = len(fx)

    l = 1

    delta = [[0 for i in range(n)] for j in range(n)]

    h = x[1] - x[0]

    s = math.floor(n / 2)

    a = x[s]

    t = (x1 - a) / h

    if abs(t) > 0.25:

        message = "не выполняется условие |t|<=0.25"

    # заполнение таблички разностей

    for i in range(n - 1):

        delta[i][0] = fx[i + 1] - fx[i]

    for i in range(1, n - 1):

        for j in range(n - i - 1):

            delta[j][i] = (delta[j + 1][i - 1] - delta[j][i - 1])

    # Расчет f (x) по формуле Стирлинга

    y1 = fx[s]

    for i in range(1, n):

        if i % 2 != 0:

            temp1 \*= (pow(t, k) - pow((k - 1), 2))

            k += 1

            d \*= i

            s = math.floor((n - i) / 2)

            y1 += (temp1 / (2 \* d)) \* (delta[s][i - 1] + delta[s - 1][i - 1])

        else:

            temp2 \*= (pow(t, 2) - pow((l - 1), 2))

            l += 1

            d \*= i

            s = math.floor((n - i) / 2)

            y1 += (temp2 / d) \* (delta[s][i - 1])

    return y1, message

# используя интерполяцию Бесселя (чет)

def ucal(u, n):

    if n == 0:

        return 1

    temp = u

    for i in range(1, int(n / 2 + 1)):

        temp = temp \* (u - i)

    for i in range(1, int(n / 2)):

        temp = temp \* (u + i)

    return temp

# расчет факториала

# заданный номер n

def fact(n):

    f = 1

    for i in range(2, n + 1):

        f \*= i

    return f

# Расчет центрального

# таблица разностей

def bessel(x, y\_vals, value):

    message = ""

    n = len(x)

    y = [[0 for i in range(n)] for j in range(n)]

    for i in range(n):

        y[i][0] = y\_vals[i]

    for i in range(1, n):

        for j in range(n - i):

            y[j][i] = y[j + 1][i - 1] - y[j][i - 1]

    # Инициализация и сумма

    summary = (y[n // 2 - 1][0] + y[n // 2][0]) / 2

    # k является источником, то есть f (0)

    if (n % 2) > 0:  # происхождение для нечетных

        k = int(n / 2)

    else:

        k = int(n / 2 - 1)  # происхождение даже

    t = (value - x[k]) / (x[1] - x[0])

    if not 0.25 <= abs(t) <= 0.75:

        message = "не выполняется условие 0.25<=|t|<= 0.75"

    # Решение по формуле Бесселя

    for i in range(1, n):

        if i % 2:

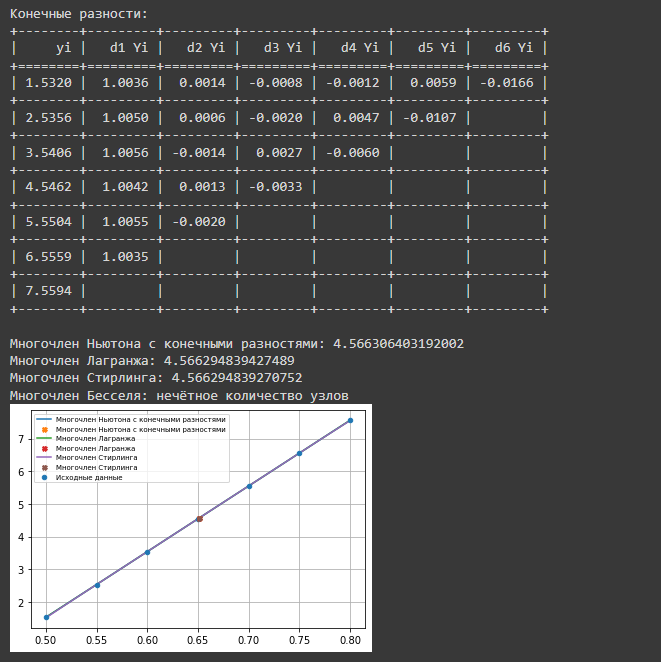
            summary = summary + ((t - 0.5) \* ucal(t, i - 1) \* y[k][i]) / fact(i)

        else:

            summary = summary + (ucal(t, i) \* (y[k][i] + y[k - 1][i]) / (fact(i) \* 2))

    return summary, message

# Пример работы программы:

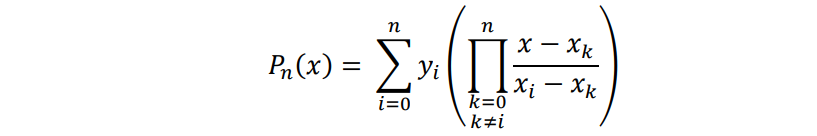


# Вывод:

Интерполяцию применяют в случае, когда требуется найти значение функции y(x) при значении аргумента xi принадлежащего интервалу [x0, … xn], но не совпадающему по значению ни с одним табличным значением этой функции. Графически задача интерполяции заключается в том, чтобы построить такую функцию, которая бы проходила через все заданные точки (узлы).

Интерполяция бывает:

* **Каноническим полиномом**. Задача интерполяции сводится к решению СЛАУ для получения коэффициентов полинома.
* **Линейная интерполяция**. Просто соединить заданные точки прямыми. Простейший метод интерполяции.
* **Интерполяция полиномом Лагранжа**.



Интерполяционный полином Лагранжа обычно применяется в теоретических исследованиях (при доказательстве теорем, аналитическом решении задач и т.п.). Минусом данного метода является то, что при добавлении точек происходит перерасчет всего многочлена.

* **Интерполяция полиномом Ньютона**. Интерполяционные формулы ньютона удобно использовать, если точка интерполяции находится в начале таблицы или в конце таблицы. Построение полинома также как и в варианте 1 сводится к определению коэффициентов ai.

Интерполяция каноническим полиномом требует бОльших вычислительных мощностей. Линейная интерполяция крайне неточная. Интерполяция методом Лагранжа хороша, если количество точек не изменяется. Интерполяция полиномом Ньютона хороша, если точки находятся в начале/конце таблицы точек.