**Министерство образования и науки Российской Федерации**

**Федеральное государственное бюджетное образовательное учреждение**

**высшего профессионального образования**

**«Московский государственный технический университет имени Н.Э. Баумана»**

**(МГТУ им. Н.Э.Баумана)**

**Мытищинский филиал**

**ФАКУЛЬТЕТ КОСМИЧЕСКИЙ**

**КАФЕДРА К-1 САУ**

**ЛАБОРАТОРНАЯ РАБОТА №7**

**ПО ДИСЦИПЛИНЕ**

**“Численные методы”**

**НА ТЕМУ:**

**«Интерполяционный многочлен Лагранжа, апроксимация методом наименьших квадратов»**

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**Студент К1-61Б**   **30.05.24 Тимофеев К. А.**

(Группа) (Подпись, дата) (ФИО)

**Руководитель**  **Чернова Т.В.**

(Подпись, дата) (ФИО)

*2024 г.*

# Теория

## Интерполяционный многочлен Лагранжа

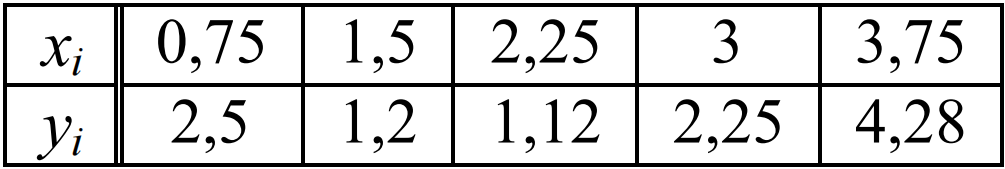
****

## Метод наименьших квадратов

****

# Задачи

1. Построить многочлен Лагранжа по следующим вводным:
2. Построить многочлен Лагранжа по варианту:
3. Аппроксимировать функции, заданные таблично, методом наименьших квадратов:



1. Аппроксимировать функции, заданные таблично по варианту, методом наименьших квадратов:



# Ход работы

Напишем функцию возвращающую значение многочлена Лагранжа в точке:

def mul(list): # поэлементное умножение векторов (П(i=0, n)(Ai))

    result = 1

    for i in list:

        result \*= i

    return result

def Lagrange(y, x, dot):

    result = 0

    lst = []

    for i in range(0, len(y)):

            lst = x[0:i]+x[i+1::1]

            result += y[i] \* mul([dot - j for j in lst]) / mul([x[i] - j for j in lst])

    return result

Вспомогательная функция отрисовки графического представления:

def drawLagrange(y, x, dots):

    lag = np.array([Lagrange(y, x, i) for i in dots])

    plt.scatter(x, y, marker='o', color="green", s = 50)

    plt.scatter(dots, lag, marker='s', color="red", s = 50)

    plt.legend((f"X:{x[:3]}\nY:{y[:3]}", f"L:{lag[:3].round(2)}"))

    plt.title("LB7 Lagrange")

    plt.grid(True)

    plt.show()

## Задание 1

x = [0.75, 1.50, 2.25, 3.00, 3.75]

y = [2.50, 1.20, 1.12, 2.25, 4.28]

u = [1.125, 1.875, 2.625, 3.375, 4.125] # точки, где ищем решение

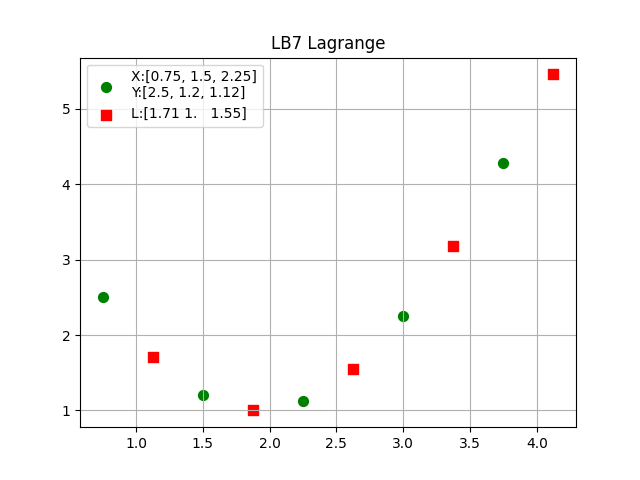
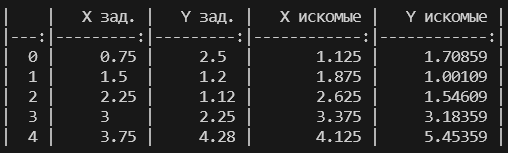
print(tabulate.tabulate(pd.DataFrame({

        "X зад.": x,  "Y зад.": y,

        "X искомые": u, "Y искомые": [Lagrange(y, x, i) for i in u]}),

        headers=["X зад.", "Y зад.", "X искомые", "Y искомые"], tablefmt='pipe'))

drawLagrange(y, x, u)



## Задание 2

x = [2.5, 8.3, 14.1, 20.5, 26.1]

y = [0.17, 0.13, 0.1, 0.13, 0.22]

u = [(x[i] + x[i-1])/2 for i in range(1, len(x))] # считаем точки между известными X

u.append(x[len(x)-1] + sum(u)/len(u)\*\*2)

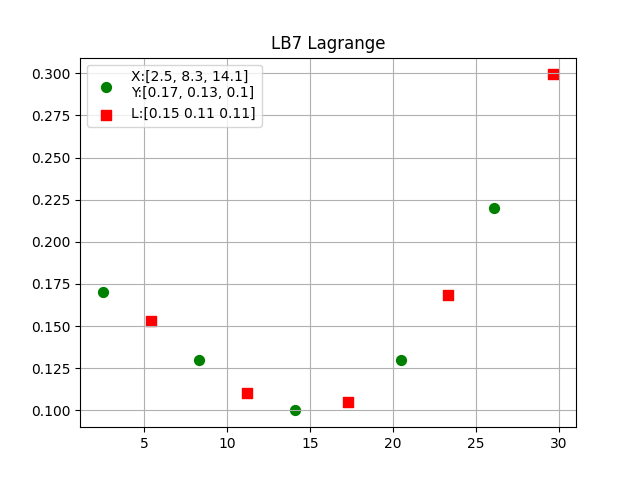
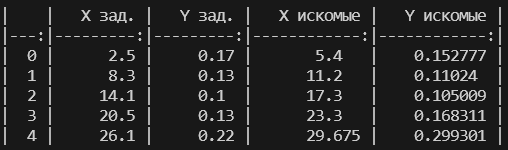
print(tabulate.tabulate(pd.DataFrame({

        "X зад.": x,  "Y зад.": y,

        "X искомые": u, "Y искомые": [Lagrange(y, x, i) for i in u]}),

        headers=["X зад.", "Y зад.", "X искомые", "Y искомые"], tablefmt='pipe'))

drawLagrange(y, x, u)



Функция поиска коэффициентов аппроксимационного многочлена по методу наименьших квадратов MMQ(…), принимает два вектора заданных значений аргумента и функции, а также тип аппроксимации. В зависимости от типа решение ищет либо функция linearSystem(..), либо quadricSystem(..) соответственно.Эти функции составляют матрицу левой половины системы и вектор правой половины, которые передаются в функцию solve(..) модуля scipy.linalg, возвращающую решение методом обратной матрицы.

def linearSystem(x, y):

    leftMatrix = np.matrix([

         [len(x), sum(x)],

         [sum(x), sum([i\*\*2 for i in x])]

    ])

    rightMatrix = np.matrix([

        [sum(y)],

        [sum([x[i]\*y[i] for i in range(0, len(x))])]

    ])

    return solve(leftMatrix, rightMatrix).reshape((2,))

def quadricSystem(x, y):

    leftMatrix = np.matrix([

         [len(x), sum(x), sum([i\*\*2 for i in x])],

         [sum(x), sum([i\*\*2 for i in x]), sum([i\*\*3 for i in x])],

         [sum([i\*\*2 for i in x]), sum([i\*\*3 for i in x]), sum([i\*\*4 for i in x])]

    ])

    rightMatrix = np.matrix([

        [sum(y)],

        [sum([x[i]\*y[i] for i in range(0, len(x))])],

        [sum([(x[i]\*\*2)\*y[i] for i in range(0, len(x))])]

    ])

    return solve(leftMatrix, rightMatrix).reshape((3,))

def MMQ(x, y, type):

    match type:

        case 'lin':

            coef = linearSystem(x, y)

            return coef

        case 'quad':

            coef = quadricSystem(x, y)

            return coef

        case \_:

            coef = linearSystem(x, y)

            return coef

Вспомогательно используем функцию отрисовки:

def drawMMQ(x, y, type):

    coef = MMQ(x, y, type)

    match type:

        case 'lin': res = [coef[0]+coef[1]\*i for i in x]

        case 'quad': res = [coef[0]+coef[1]\*i+coef[2]\*i\*\*2 for i in x]

        case \_: res = [coef[0]+coef[1]\*i for i in x]

    dt = pd.DataFrame({

        "Xi": x, "Yi": y,

        f"МНК {type}":res,

        "Расхождение": [abs(y[i]-res[i]) for i in range(0, len(y))]})

print(tabulate.tabulate(dt,headers='keys', tablefmt='psql', showindex=False))

    plt.scatter(x, y, marker='o', color='red')

    plt.plot(x, res, color='green')

    plt.scatter(x, res, color='green', marker='s')

    plt.title(f"LB7 МНК ({type})")

    plt.legend(("Заданно", "МНК", "МНК"))

    plt.show()

## Задание 3

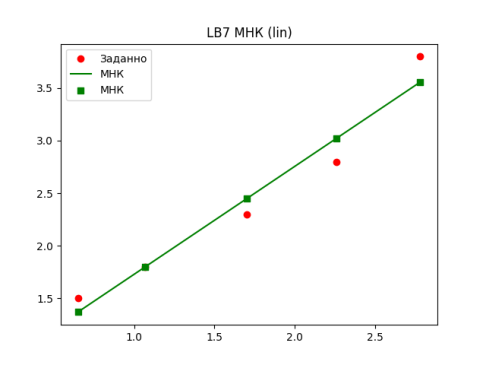
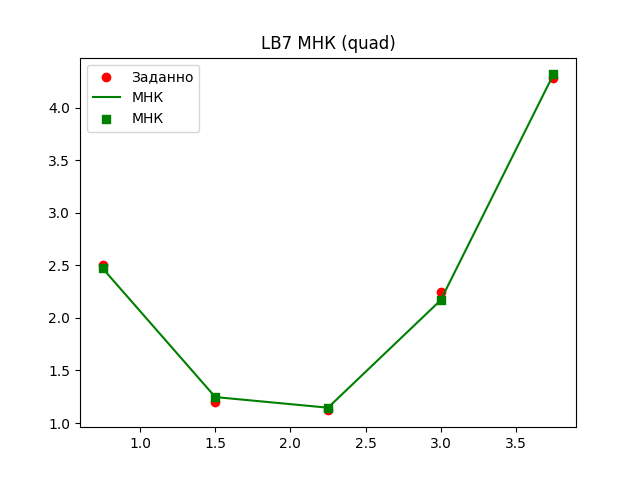
drawMMQ([0.65, 1.07, 1.7, 2.26, 2.78],

        [1.5, 1.8, 2.3, 2.8, 3.8], 'lin')

drawMMQ([0.75, 1.5, 2.25, 3.0, 3.75],

        [2.5, 1.2, 1.12, 2.25, 4.28], 'quad')

## Задание 4



drawMMQ([2.5, 8.3, 14.1, 20.5, 26.1],

        [0.17, 0.13, 0.1, 0.13, 0.22], 'lin')

drawMMQ([2.5, 8.3, 14.1, 20.5, 26.1],

        [0.17, 0.13, 0.1, 0.13, 0.22], 'quad')

