**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 №4**

**ПО ДИСЦИПЛИНЕ**

**“Численные методы”**

**НА ТЕМУ:**

**«Решение нелинейных уравнений методами Ньютона (касательных) и комбинированным методом хорд и касательных»**

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

**Студент К1-61Б**  **20.04.24 Тимофеев К. А.**

(Группа) (Подпись, дата) (ФИО)

**Руководитель**  **Чернова Т.В.**

(Подпись, дата) (ФИО)

*2024 г.*

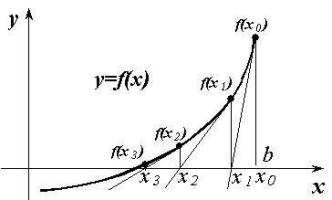
# Задачи

1. Найти корень уравнения ; методом касательных.
2. Найти второй корень уравнения ; на отрезке  методом касательных.
3. Найти корень уравнения из приложения по своему варианту (12) ; методом касательных.
4. Найти корень уравнения  комбинированным методом касательных и хорд.
5. Найти корень уравнения из приложения по своему варианту (12) ; комбинированным методом касательных и хорд.

# Теория

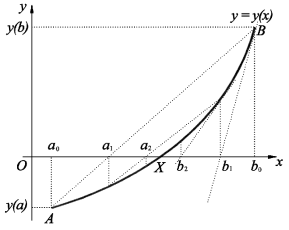
* Метод Ньютона





* Комбинированный метод хорд и касательных (далее КХК)





# Ход работы

## Реализация

Напишем функцию для поиска решения нелинейного уравнения

методом Ньютона.

def Newton(function, start, epsilon):

    currentX: list[float] = [start]

    newX: list[float] = []; epsControl: list[float] = []; zeroControl: list[float] = []

    iter: list[int] = []

    i = 0

    while True:

        i+=1                                    # f(x) / df(x)dx

        newX.append(currentX[len(currentX)-1] - function(currentX[len(currentX)-1]) / dfdx(function, currentX[len(currentX)-1], epsilon))

        zeroControl.append(function(newX[len(newX)-1]))

        iter.append(i)

        if abs(newX[len(newX)-1] - currentX[len(currentX)-1]) < epsilon:

            epsControl.append(f"Root={newX[len(newX)-1]}")

            Break

        epsControl.append(abs(currentX[len(currentX)-1] - newX[len(newX)-1]))

        currentX.append(newX[len(newX)-1])

    return pd.DataFrame({

        "Xn": currentX,

        "Xn+1": newX,

        "Оценка погрешности": epsControl,

        "Контроль нуля Y(Xn+1)": zeroControl,

        "Число итераций": iter

    })

Способствующие функции

def dfdx(function, x, epsilon):

    return 0.5\*(function(x + epsilon)-function(x - epsilon))/epsilon

def d2fdx2(function, x, epsilon):

    return (function(x-epsilon)-2\*function(x)+function(x+epsilon))/epsilon\*\*2

Также напишем функцию для поиска решения методом КХК

def chordsTangents(function, a0, b0, epsilon):

    a : list[float] = [a0]; b : list[float] = [b0]

    fa : list[float] = [function(a0)]; fb : list[float] = [function(b0)]

    dfda : list[float] = [dfdx(function, a0, epsilon)]; dfdb : list[float] = [dfdx(function, b0, epsilon)]

    epsControl : list[float] = [abs(le(b)-le(a))]; left : bool = True

    if le(fa)\*le(fb)>0.0:

        print(f"#--No roots--\nfa = {fa};\nfb = {fb};------------#")

        return None

    elif le(fa)\*le(fb)==0.0:

        print("#--fa\*fb==0.0--")

    while True:

        if le(fa)\*dfdx(function, le(dfda), epsilon) > 0.0:

            # tangents - left, chords - right

            a.append(le(a)-le(fa)/le(dfda))

            b.append((le(a)\*le(fb)-le(b)\*le(fa))/(le(fb)-le(fa)))

            fa.append(function(le(a)))

            dfda.append(dfdx(function, le(a), epsilon))

            fb.append(function(le(b)))

            dfdb.append(dfdx(function, le(b), epsilon))

            left = True

        elif le(fb)\*dfdx(function, le(dfdb), epsilon) > 0.0:

            # tangents - right, chords - left

            b.append(le(b)-le(fb)/le(dfdb))

            a.append((le(a)\*le(fb)-le(b)\*le(fa))/(le(fb)-le(fa)))

            fa.append(function(le(a)))

            dfda.append(dfdx(function, le(a), epsilon))

            fb.append(function(le(b)))

            dfdb.append(dfdx(function, le(b), epsilon))

            left = False

        if abs(le(b)-le(a))<epsilon:

            epsControl.append(f'Корень = {(le(b)+le(a))/2}')

            break

        epsControl.append(abs(le(b)-le(a)))

    if left:

        return pd.DataFrame({

            "a":a, "b": b,

            "f(a)": fa, "f(b)":fb,

            "df(a)dx": dfda,

            "Оценка погрешности": epsControl

        })

    else:

        return pd.DataFrame({

            "a":a, "b": b,

            "f(a)": fa, "f(b)":fb,

            "df(b)dx": dfdb,

            "Оценка погрешности": epsControl

        })

Взятие последнего элемента списка реализовано через функцию le(arr) \*last element

def le(x): # get last element

    return x[len(x)-1]

## Задание 1

# TASK 1: x^2 - sqrt(x+4) = 0; [1, 2] eps = 0.001

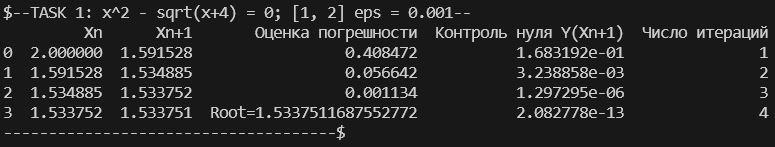
x0 = 2.0

eps = 0.001

def f(x):

return x\*\*2 - np.sqrt(x+4)

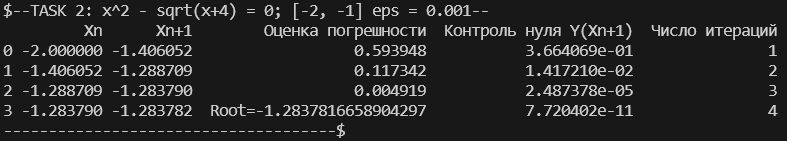
print(f'\n$--TASK 1: x^2 - sqrt(x+4) = 0; [1, 2] eps = 0.001--\n{Newton(f, x0, eps)}\n-------------------------------------$')



## Задание 2

# TASK 2: x^2 - sqrt(x+4) = 0; [-2, -1] eps = 0.001

print(f'\n$--TASK 2: x^2 - sqrt(x+4) = 0; [-2, -1] eps = 0.001--\n{Newton(f, -2, eps)}\n-------------------------------------$')



## Задание 3

# TASK 3: x^3 + 3\*x^2 - 1 = 0; eps = 0.001

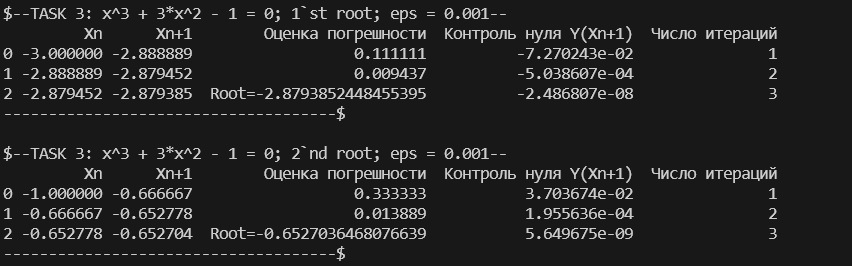
def f1(x):

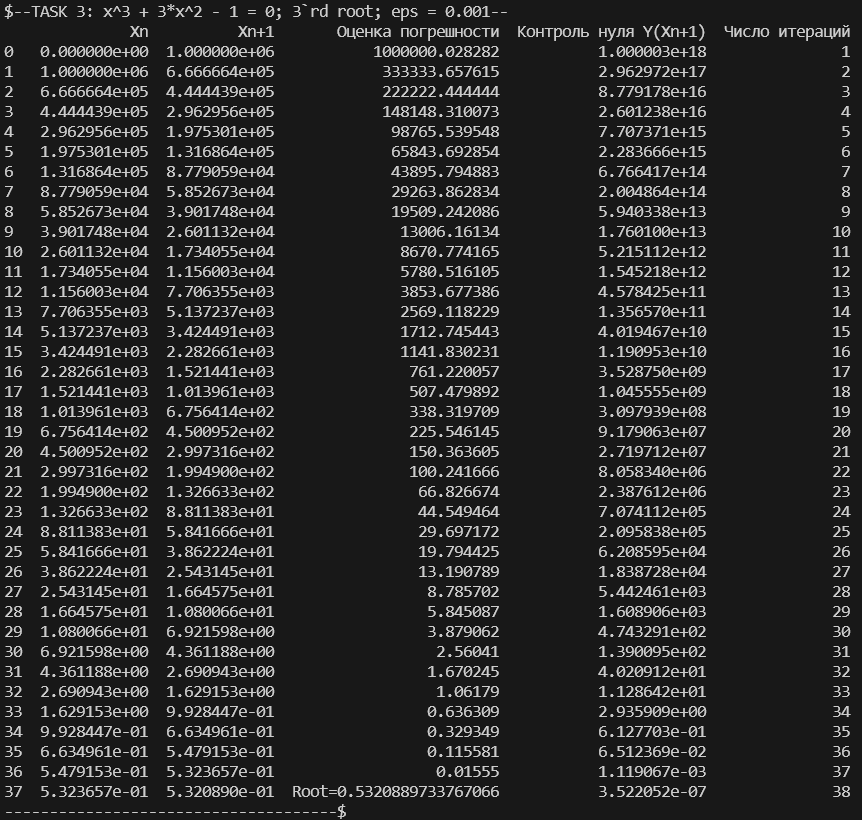
    return x\*\*3 + 3\*x\*\*2 - 1

print(f'\n$--TASK 3: x^3 + 3\*x^2 - 1 = 0; 1`st root; eps = 0.001--\n{Newton(f1, -3, eps)}\n-------------------------------------$')

print(f'\n$--TASK 3: x^3 + 3\*x^2 - 1 = 0; 2`nd root; eps = 0.001--\n{Newton(f1, -1, eps)}\n-------------------------------------$')

print(f'\n$--TASK 3: x^3 + 3\*x^2 - 1 = 0; 3`rd root; eps = 0.001--\n{Newton(f1, 0, eps)}\n-------------------------------------$')





## Задание 4

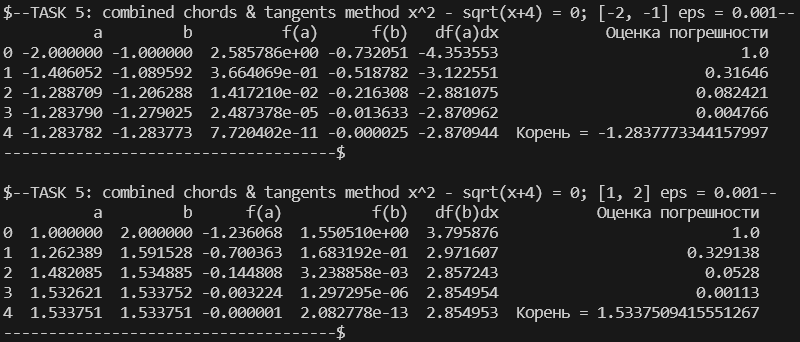
# TASK 5: combined chords & tangents method

print(f'\n$--TASK 5: combined chords & tangents method x^2 - sqrt(x+4) = 0; [-2, -1] eps = 0.001--\n \

      {chordsTangents(f, -2.0, -1.0, eps)}\n-------------------------------------$')

print(f'\n$--TASK 5: combined chords & tangents method x^2 - sqrt(x+4) = 0; [1, 2] eps = 0.001--\n \

      {chordsTangents(f, 1.0, 2.0, eps)}\n-------------------------------------$')

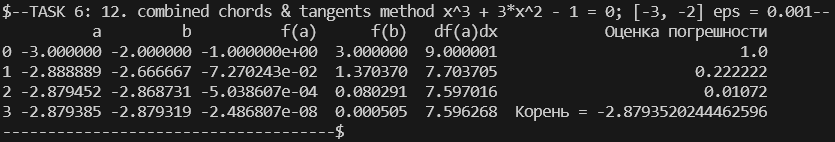


## Задание 5

# TASK 6: 12. combined chords & tangents method

print(f'\n$--TASK 6: 12. combined chords & tangents method x^3 + 3\*x^2 - 1 = 0; [-3, -2] eps = 0.001--\n \

      {chordsTangents(f1, -3.0, -2.0, eps)}\n-------------------------------------$')



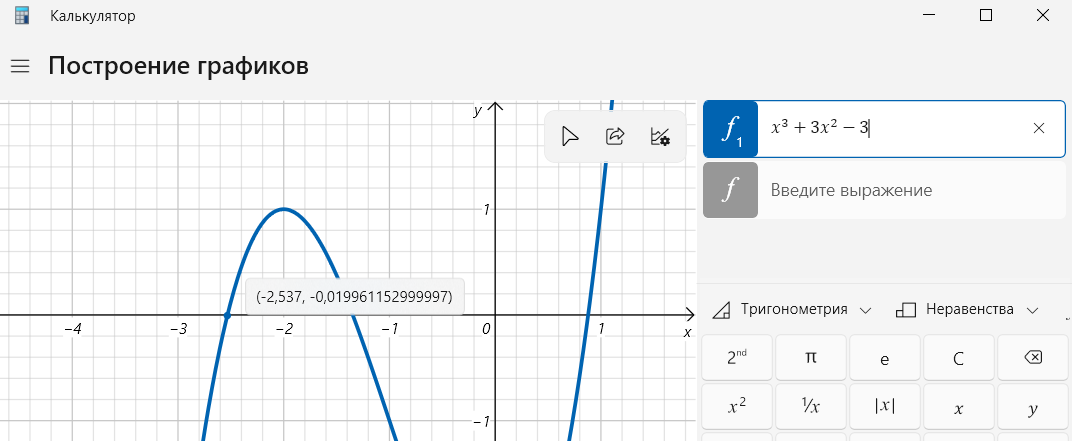
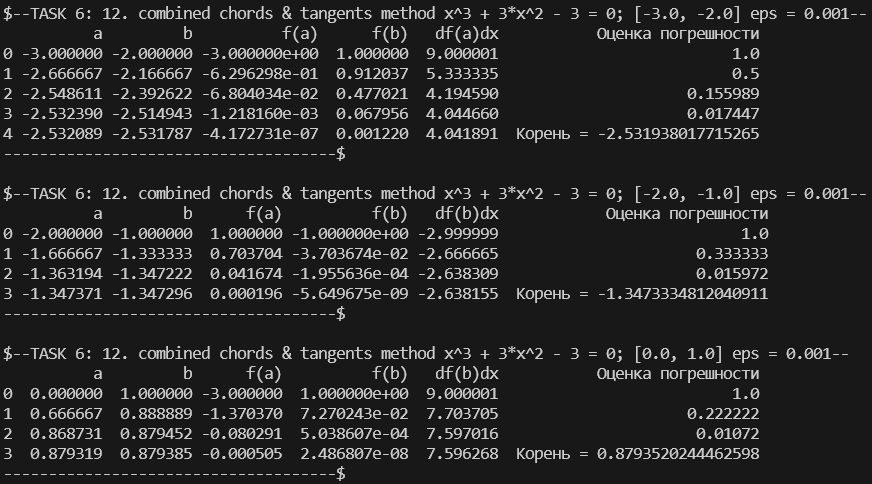
## Выводы

Изучены два метода решения нелинейных уравнений, а также построены алгоритмы их реализации. Программы работают корректно, за исключением одного нюанса.

На достаточно близких к друг другу корнях  алгоритм может вести себя некорректно, особенно на корнях в области нуля. Так для варианта 12 из приложения к лабораторной работе КХК метод не выходит из цикла вычислений до момента переполнения памяти, выделенной для программы.

Решением этой проблемы, является поиск достаточно малого отрезка [a, b], достаточного для эффективного решения.

Например для функции

на отрезках [-3, -2], [-2, -1], [0, 1], алгоритм работает штатно.