МИНИСТЕРСТВО ЦИФРОВОГО РАЗВИТИЯ, СВЯЗИ И МАССОВЫХ КОММУНИКАЦИЙ РОССИЙСКОЙ ФЕДЕРАЦИИ

Ордена Трудового Красного Знамени федеральное государственное бюджетное образовательное учреждение высшего образования

**«Московский технический университет связи и информатики»**

Кафедра «Информатика»

**Отчет по заданию №1**

**по дисциплине**

**«Численные методы»**

Выполнил: студент гр. БЭИ2202

Кулешов А. С.

Вариант 16.

Проверил: доц. каф. «Информатика»

Мацкевич А. Г.

Москва, 2023 г.

**1. Индивидуальное задание**

Решить уравнение при помощи вспомогательных методов: метода Ньютона и метода хорд. Для 2-го заданного метода выполнить решение уравнения с точностью 10-4, создав программу, реализующую заданный метод.

**2. Этап отделения корней**

Используем для этого ЯП Python. Отделение корней произведем графическим методом (график функции) при помощи библиотеки Plotly.

**import** pandas **as** pd

**import** numpy **as** np

**import** plotly.graph\_objects **as** go

**from** sympy **import** \*

**def** f(x):

**return** np.sin(1 – 0.2\*x\*x) - x

begin = -2

end = 7

step = 0.2

size = 1000

x = np.linspace(begin, end, size)

y\_f = f(x)

fig = go.Figure() # layout\_xaxis\_range = [-2,6]

fig.add\_trace(go.Scatter(x=[begin, end],

y=[0, 0],

name="Ось Х"))

fig.add\_trace(go.Scatter(x=x,

y=y\_f,

name="График функции"))

fig.update\_layout(title="Построение графика функции",

xaxis\_title="Значение аргумента",

yaxis\_title="Значение функции",

margin=dict(l=10, r=70, t=50, b=0))

fig.show()

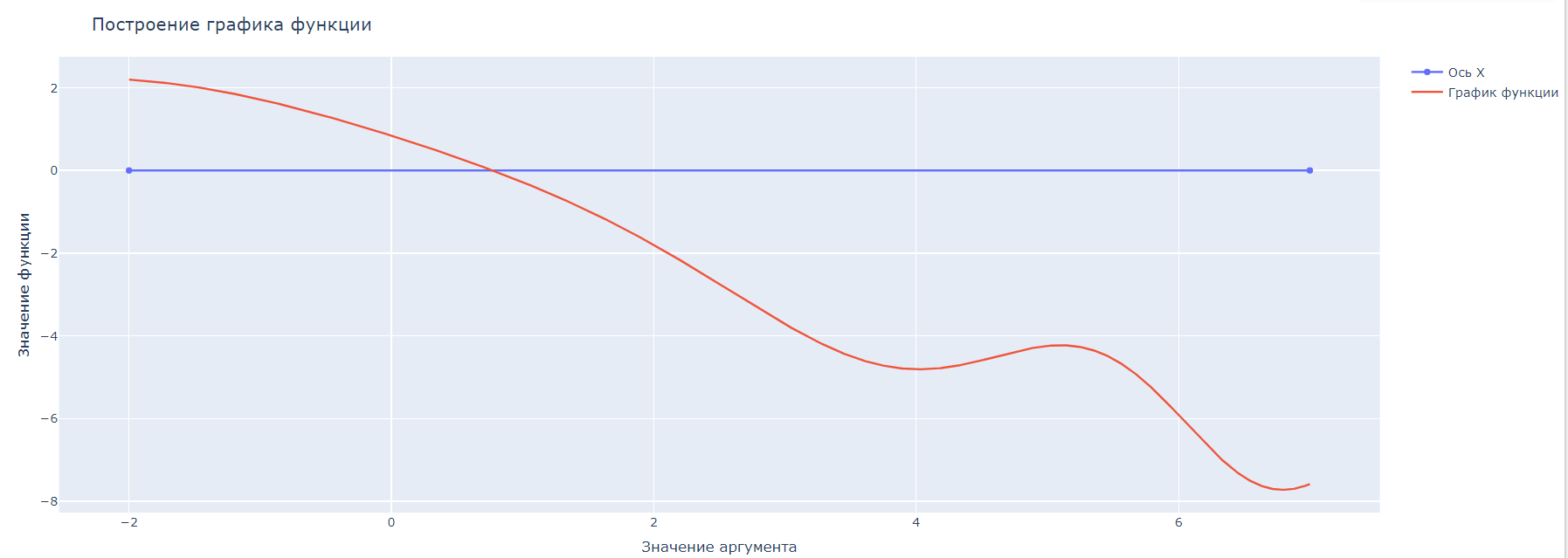


Рисунок 1 – Графическое отделение корня уравнения

Из построенного графика функции f(x) видно, что на отрезке [0, 2] есть один корень. На этом графический способ отделения корней заканчивается.

Теперь найдём отрезок с корнем аналитически, см. рис.2

Для аналитического отделения корня построена таблица рис.2.

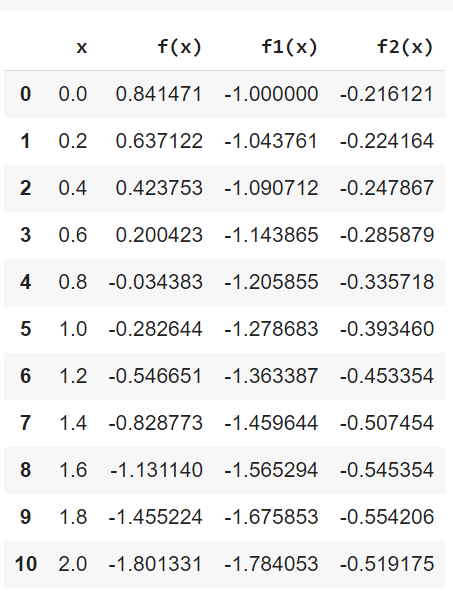


Рисунок 2 – Таблица значений функции и ее производных

В столбцах таблицы выведены некоторые значения аргумента x на заданном отрезке, а также значения функций при этих значениях x.

Видно, что на отрезке [0, 2] функция f(x) меняет знак, значит существует, по крайней мере, один корень.

Значения первой и второй производных в заданных точках отрезка [0, 2] не меняет знак, однако этой информации недостаточно для того, чтобы с уверенностью сказать, что производные не меняют знак, поэтому найдём производные: , анализировав функции можно сделать вывод, что они обе меньше 0 на отрезке [0;2].

**3. Этап уточнения корня**

**3.1 Метод Ньютона**

Необходимые и достаточные условия сходимости метода Ньютона:

* f(x) непрерывна на [a; b] и ;
* f(x) и f(x) отличны от нуля и сохраняют знаки для.

В нашем случае на отрезке [0;2] требования сходимости выполняются.

Начальное приближение аргумента x0 должно удовлетворять условию:, т. е. за начальное приближение следует принять тот конец отрезка, где знак функции и знак второй производной совпадают. Поскольку и , то выберем начальное приближение к корню: x0=2.

Для получения решения уравнения методом Ньютона воспользуемся следующей рекуррентной формулой: .

В нашем случае , x0 = 2.

E = [i **for** i **in** range(-2, -5, -1)]

sp\_i = []

sp\_x = []

sp\_f = []

def newton\_method(f, f\_prime, x0, tolerance=1e-6, max\_iterations=4):

    x = x0

    for i in range(max\_iterations):

        sp\_i.append(i)

        sp\_x.append(x)

        sp\_f.append(f(x))

        fx = f(x)

        if abs(fx) < tolerance:

            return x

        fpx = f\_prime(x)

        if fpx == 0:

            break

        x = x - fx / fpx

    return x

def f(x):

    return sin(1 - 0.2\*x\*x) - x

def f\_prime(x):

    return -1 - 0.4\*x\*cos(1 - 0.2\*x\*x)

root = newton\_method(f, f\_prime, x0=2)

df = pd.DataFrame(data={'n':range(sp\_i[-1]+1), 'x': sp\_x, 'f(x)': sp\_f})

df

Результат вычисления корня тремя иттерациями представлен на рис. 3

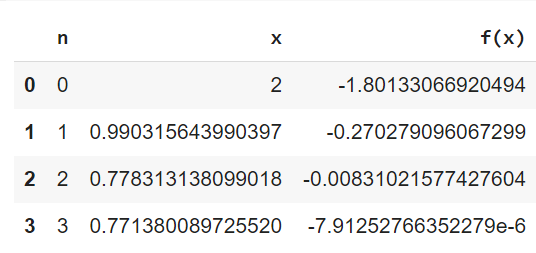


Рисунок 3 – Результаты вычисления при помощи метода Ньютона.

Оценим погрешность после трёх итераций:

**3.2 Метод хорд**

Необходимые и достаточные условия сходимости метода хорд аналогичны методу Ньютона:

* f(x) непрерывна на [a; b] и ;
* f(x) и f(x) отличны от нуля и сохраняют знаки для.

Соответственно, т.к. условия совпадают с условиями метода Ньютона, то перейдём к вычислениям:

sp\_i = []

sp\_x = []

sp\_f = []

def chord\_method(f, a, b, tolerance=1e-4, max\_iterations=100):

    fa = f(a)

    fb = f(b)

    с = -1

    for i in range(max\_iterations):

        if abs(fa - fb) < tolerance:

            return (a + b) / 2

        c = (a \* fb - b \* fa) / (fb - fa)

        fc = f(c)

        sp\_i.append(i)

        sp\_x.append(с)

        sp\_f.append(fc)

        if abs(fc) < tolerance:

            return c

        if fc \* fa < 0:

            b = c

            fb = fc

        else:

            a = c

            fa = fc

    return с

def f(x):

    return sin(1 - 0.2\*x\*x) - x

root = chord\_method(f, a=0, b=2)

df = pd.DataFrame(data={'n':range(sp\_i[-1]+1), 'x': sp\_x, 'f(x)': sp\_f})

df

Результат вычисления до точности представлен на рисунке 4.

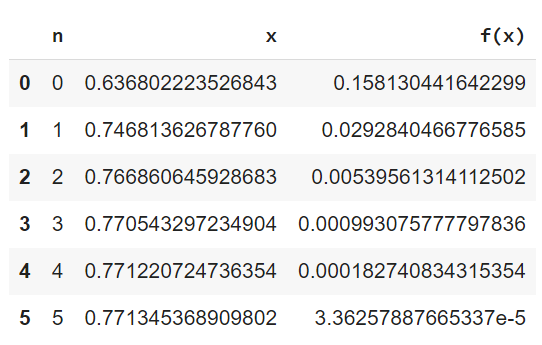


Рисунок 4 – Результаты вычисления при помощи метода хорд.

Оценим погрешность после пяти итераций:

График зависимости количества итераций от погрешности представлен на рисунке 5.

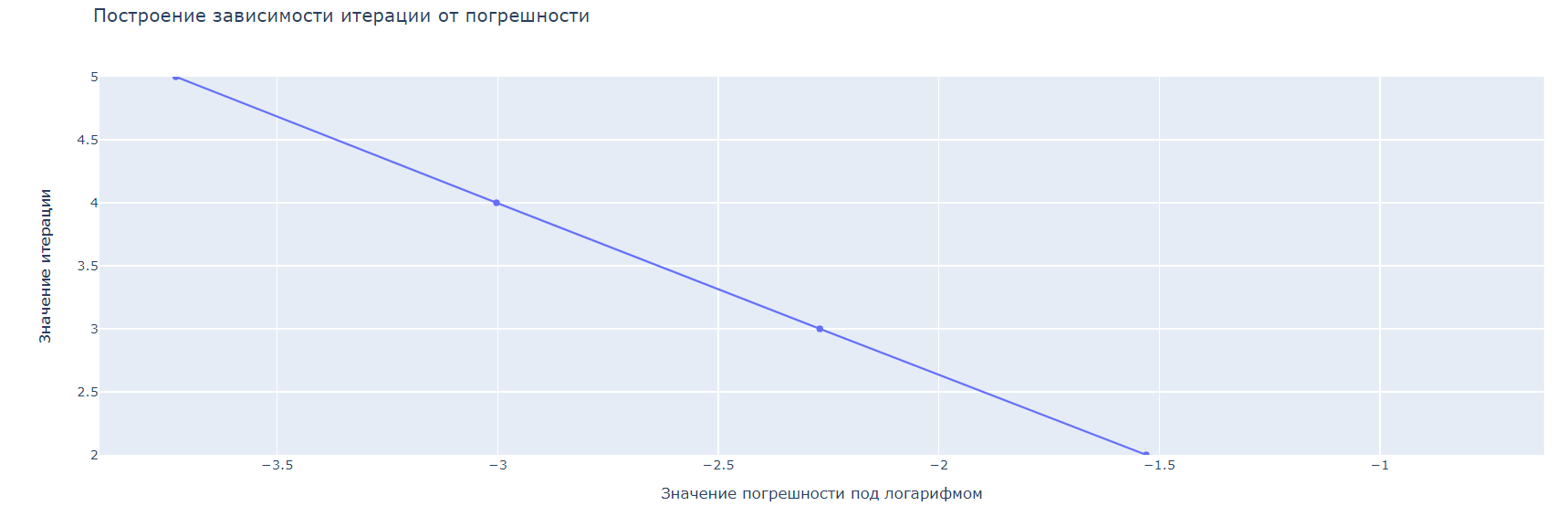


Рисунок 5 – Зависимости количества итераций от погрешности

**4. Заключение**

Сравнивая метод хорд и метод Ньютона, можно с уверенностью сказать, что при помощи метода Ньютона приближенное значение х сходится к корню уравнения быстрее.