**Министерство науки и высшего образования Российской Федерации**
**Федеральное государственное бюджетное образовательное учреждение высшего образования**
**сМосковский государственный технический университет**
**имени Н.Э. Баумана**
**(национальный исследовательский университет)»**
**(МГТУ им. Н.Э. Баумана)**

| | |
|---|---|
| ФАКУЛЬТЕТ | Информатика и системы управления |
| Кафедра | ИУ6 Компьютерные системы и сети |
| Группа | ИУ6-63Б |

# ВЫЧИСЛИТЕЛЬНАЯ МАТЕМАТИКА

## Отчет по домашнему заданию 2:

## Прямые методы решения СЛАУ

## Вариант 10

| | | |
|---|---|---|
| Студент: | | В.К. Залыгин |
| | дата, подпись | Ф.И.О. |
| Преподаватель: | | Я.Ю. Павловский |
| | дата, подпись | Ф.И.О. |

*Москва, 2025*

## СОДЕРЖАНИЕ

# ВВЕДЕНИЕ

Цель домашней работы: изучение методов половинного деления (бисекций), метода простой итерации, метода Ньютона для решения нелинейных уравнений.

Исходя из задания, необходимо с использованием графического калькулятора определить отрезок локализации для корня уравнения, реализовать указанные методы решения (метод биекция, метод простой итерации и метод Ньютона) нелинейных уравнений на определенном отрезке локализации с заданной точностью и провести решение 2 заданных задач указанными методами. Вариант для выполнения представлен на рисунке 1.

***Вариант 10***

1. **Отделить корни уравнения и найти его методом половинного деления с точностью** $\varepsilon = 0{,}001$ $2^x - 3x - 2 = 0$
2. **Отделить корни уравнения и найти его методом простой итерации и методом Ньютона с точностью** $\varepsilon = 0{,}001$. $tg(0{,}5x - 1{,}2) = x^2 - 1$

Рисунок 1 – Заданные задачи по варианту

## Краткая теоретическая справка

**Метод половинного деления (бисекций)** – алгоритм для нахождения корня уравнения f(x)=0 на отрезке [a,b], где f(x) имеет противоположные знаки на концах. Суть метода: интервал делится пополам, и дальнейший поиск ведётся в той половине, где сохраняется смена знака функции. Сходимость гарантирована при непрерывности f(x) и выполняемости $f(a) \cdot f(b) < 0$. Работа метода завершается, когда половина длины интервала сокращается до заданной точности ε $(b-a)/2 < \varepsilon$

**Метод простой итерации** — подход, основанный на преобразовании уравнения f(x)=0 к виду x=ϕ(x). На каждом шаге новое приближение вычисляется как $x_{n+1}=\phi(x_n)$. Условие сходимости заключается в том, что производная ϕ′(x) должна удовлетворять $|\phi'(x)|<1$ вблизи корня, чтобы итерации не расходились. Остановка происходит при достижении малой разницы между $x_{n+1}$ и $x_n$ или малого значения $|f(x_n)|$.

**Метод Ньютона** — быстрый алгоритм, использующий производную для построения касательных к f(x). На каждом шаге приближение уточняется по формуле $x_{n+1}=x_n-f(x_n)/f'(x_n)$. Для сходимости требуется, чтобы начальное x0 было близко к корню, а производная f′(x) не обращалась в ноль. Алгоритм завершается, когда изменения между итерациями или значение функции становятся меньше ε.

## Графики с локализованным корнем

На рисунках 2 и 3 представлены графики с локализированным корнем для заданных уравнений.

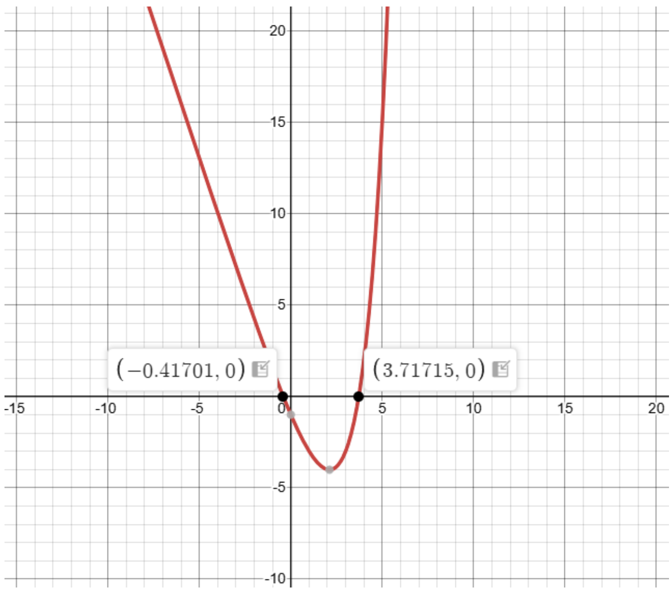


Рисунок 2 – График уравнения 2^x – 3x - 2

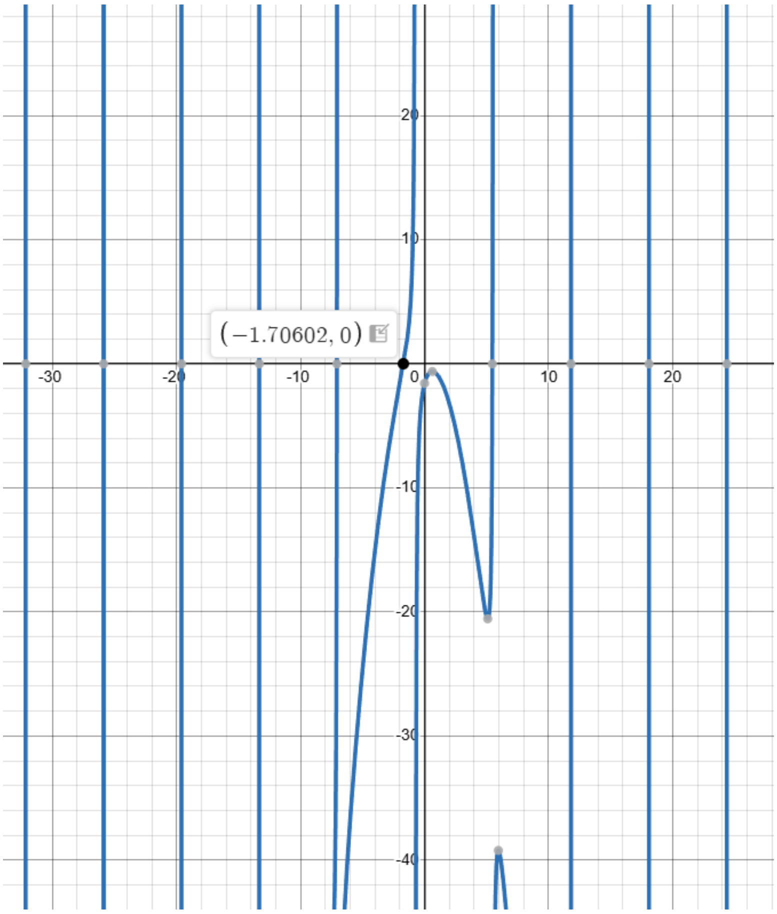


Рисунок 3 – График уравнения tg(0,5x – 1.2) – x^2 + 1

**Программы для решения**

В листинге 1 и 2 представлен код для решения уравнений 2^x – 3x - 2= 0 и tg(0,5x – 1.2) – x^2 + 1=0 из варианта на язык python.

Листинг 1 – Код для решения первого уравнения

```
def bisection(f, a, b, eps, max_iter=1000):
    if f(a) * f(b) >= 0:
        raise ValueError(f"Функция не меняет знак на интервале
[{a}, {b}]")

    iterations = 0
    while iterations < max_iter:
        c = (a + b) / 2
        if (b - a)/2 < eps:  # Основной критерий остановки
            return c, iterations + 1
        if f(a) * f(c) < 0:
            b = c
        else:
            a = c
        iterations += 1

    raise RuntimeError(f"Достигнут предел итераций ({max_iter})")

def cubic_eq(x):
    return x**3 - 6*x - 7

if __name__ == "__main__":
```

```
    print("Задача 1: Метод бисекций")
    x_bisect, it_bisect = bisection(f1, -5, 0, 0.001)
    print(f"Корень: {x_bisect:.5f}, итерации: {it_bisect}")
    x_bisect, it_bisect = bisection(f1, 0, 5, 0.001)
    print(f"Корень: {x_bisect:.5f}, итерации: {it_bisect}")

    except Exception as e:
        print(f"Ошибка: {str(e)}")
```

Листинг 2 – Код для решения второго уравнения

```
def simple_iteration(phi, f, x0, eps, max_iter=1000):
    x_prev = x0
    for iterations in range(1, max_iter + 1):
        x_next = phi(x_prev)
        # Комбинированный критерий остановки
        if abs(x_next - x_prev) < eps and abs(f(x_next)) < eps:
            return x_next, iterations
        x_prev = x_next
    raise RuntimeError(f"Не сошлось за {max_iter} итераций")

def newton(f, df, x0, eps, max_iter=1000):
    x = x0
    for iterations in range(1, max_iter + 1):
        dfx = df(x)
        if abs(dfx) < 1e-12:
            raise ZeroDivisionError("Производная близка к нулю")
        x_new = x - f(x)/dfx
        if abs(x_new - x) < eps:
            return x_new, iterations
        x = x_new
    raise RuntimeError(f"Не сошлось за {max_iter} итераций")

# Уравнение x³ - 3x² + 4x - 2 = 0
def f(x):
    return tan(0.5*x - 1.2) - x**2 + 1



def df(x):
    return 1 / ( 2 * cos (0.5*x - 1.2)**2) - 2*x



def phi(x):
    return x - 0.2*f2(x)

if __name__ == "__main__":
    eps = 0.001
    print("="*50)
    print("="*50)
    # Метод простой итерации и Ньютона
    try:
        print("\nЗадача 2: Метод простых итераций")
        x_iter, it_iter = simple_iteration(phi2, -1.0, 0.0001)
        print(f"Корень: {x_iter:.5f}, итерации: {it_iter}")
```

```
        print("\nЗадача 2: Метод Ньютона")
        x_newt, it_newt = newton(f2, df2, 2.0, 0.0001)
        print(f"Корень: {x_newt:.5f}, итерации: {it_newt}")
    except Exception as e:
        print(f"\n✗ Ошибка: {str(e)}")
```

## Результат выполнения программы

Результаты выполнения программ представлены на рисунке 4 и в таблицах 1 и 2 для уравнений 2^x – 3x - 2= 0 и tg(0,5x – 1.2) – x^2 + 1=0 соответственно. Результаты получены с заданной точностью.

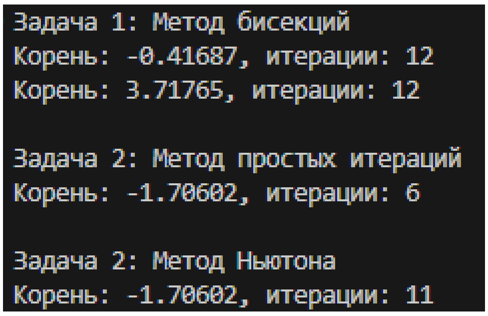


Рисунок 4 – Результат выполнения программы для поиска решений уравнений

Таблица 1 – Результаты вычисления 2^x – 3x - 2= 0

| Метод решения | Приближённый корень | Количество итераций |
|---|---|---|
| Метод бисекций | ≈ 0.41687 | 12 |
| | ≈ 3.71765 | 12 |

Таблица 2 – Результаты вычисления tg(0,5x – 1.2) – x^2 + 1= 0

| Метод решения | Приближённый корень | Количество итераций |
|---|---|---|
| Метод простых итераций | ≈ -1.70602 | 6 |
| Метод Ньютона | ≈ -1.70602 | 11 |

## Анализ результатов

Решения всеми методами обоих уравнений дали корректные результаты, соответствующие заданной точности. Исходя из данных таблицы 2, можем увидеть, что метод простых итераций сходится быстрее метода Ньютона (полученные результаты при подсчете количества итераций 6 и 11 соответственно).