Министерство науки и высшего образования Российской Федерации

Федеральное государственное бюджетное образовательное учреждение высшего образования

«Рязанский государственный радиотехнический университет имени В.Ф. Уткина»

Кафедра «Вычислительная и прикладная математика»

**Отчет по лабораторной работе № 3**

Решение систем линейных алгебраических уравнений

по дисциплине: «Вычислительная математика»

Выполнил:  
студент группы 143  
Попов К.И.  
Проверил:   
доц. каф. ВПМ  
Крошилина С.В.

# Рязань 2024

# 1 Задание (Вариант 2)

Для заданного варианта (Рисунок 1) решить методом итераций систему уравнений A⋅X = B. Для остановки процесса последовательных приближений использовать условие: сумма модулей приращений элементов вектора X на последнем шаге итераций меньше ε = 0.001. Используя метод Гаусса, вычислить определитель и число обусловленности матрицы A.

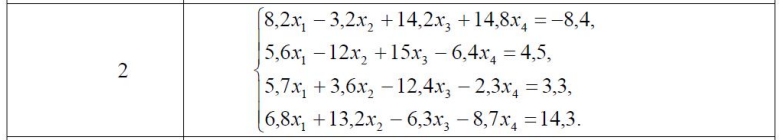


Рисунок – Вариант задания

# 2 Блок-схемы алгоритмов

## 2.1 Метод итераций

Схема метода итераций представлена на рисунке 2.

Рисунок – Схема метода итераций

## 2.2 Метод Гаусса

Схема метода дихотомии представлена на рисунке 3.

Рисунок – Схема метода Гаусса

# 3 Листинг алгоритмов

Реализация алгоритмов приведена на языке Python.

## 3.1 Листинг метода итераций

def alpha\_coef(self, A, length):

        alpha = np.array([[0.0 for \_ in range(length)] for \_ in range(length)])

        for i in range(length):

            for j in range(length):

                if j != i:

                    alpha[i, j] = -A[i, j] / A[i, i]

        return alpha

    def beta\_coef(self, A, B, length):

        beta = np.array([[0.0] for \_ in range(length)])

        for i in range(length):

            beta[i] = B[i] / A[i, i]

        return beta

    def iterations\_method(self, alpha, beta, eps=1e-4, max\_iterations=1000):

        x = np.zeros\_like(beta)

        x\_prev = x.copy()

        for \_ in range(max\_iterations):

            for i in range(len(x)):

                summary = 0

                for j in range(len(x)):

                    if i != j:

                        summary += alpha[i, j] \* x[j]

                x[i] = summary + beta[i]

            if np.abs(x - x\_prev).all() < eps:

                return x

            x\_prev = x.copy()

        return x

## 3.2 Листинг метода Гаусса

def lu\_decomposition(self, matrix):

        n = len(matrix)

        L = np.eye(n)

        U = np.zeros((n, n))

        for i in range(n):

            for k in range(i, n):

                U[i][k] = matrix[i][k] - np.dot(L[i,:i], U[:i,k])

            for k in range(i + 1, n):

                L[k][i] = (matrix[k][i] - np.dot(L[k,:i], U[:i,i])) / U[i][i]

        return U

    def determinant(self, matrix):

        U = self.lu\_decomposition(matrix)

        det = np.prod(np.diag(U))

        return det

    def condition(self, matrix):

        inv\_matrix = np.linalg.inv(matrix)

        return np.max(np.sum(np.abs(matrix), axis=1)) \* np.max(np.sum(np.abs(inv\_matrix), axis=1))

# 4 Пользовательский интерфейс

На рисунке 7 показано главное пользовательское окно взаимодействия с программой, в котором пользователь может получить корень уравнения выбранным методом.

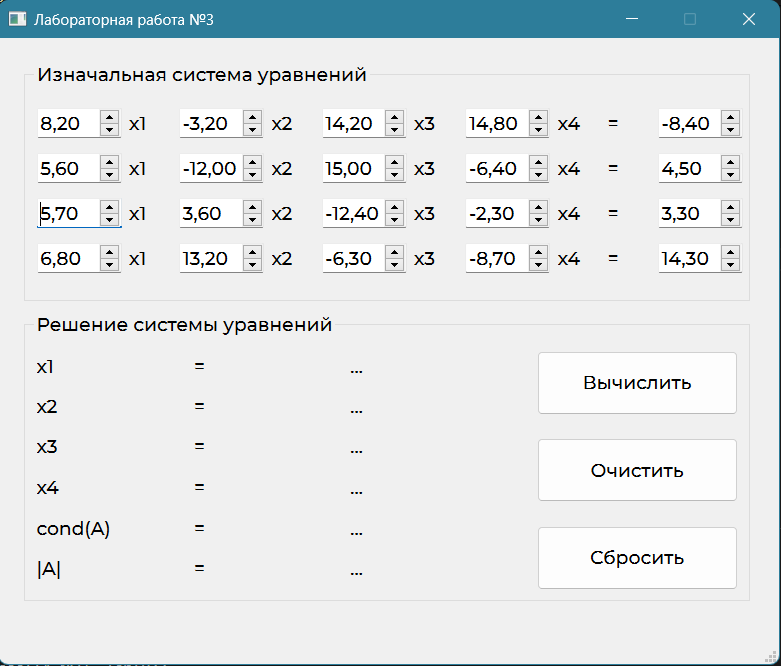


Рисунок – Главное окно

# 5 Результат работы программы

Ответ, посчитанный методом дихотомии, представлен на рисунке 8.

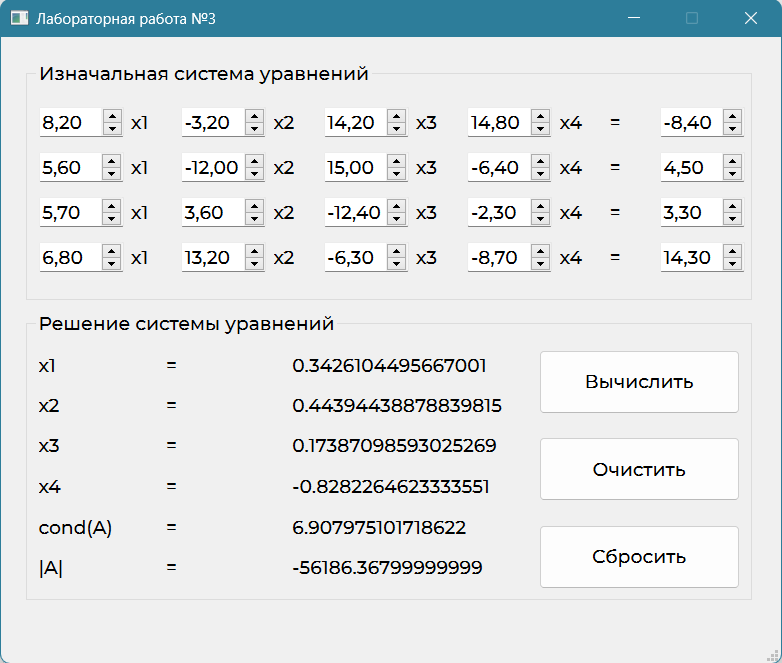


Рисунок – Результат работы метода дихотомии

# 6 Проверка полученного результат

Вид матрицы в треугольном виде и ее корни представлены на рисунке 14.

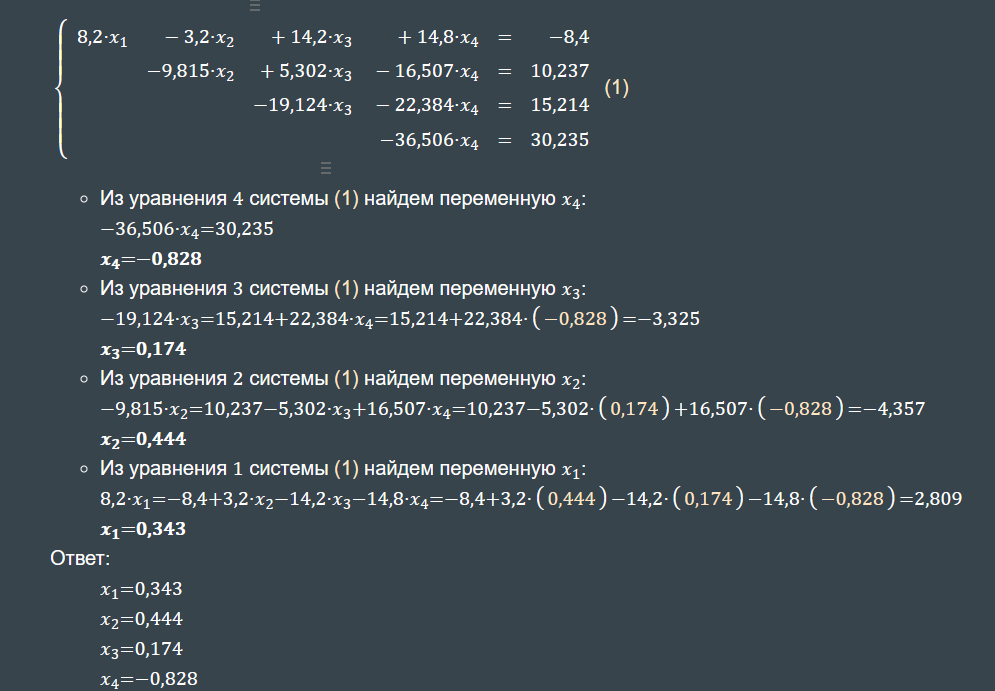


Рисунок – График заданной функции

Определитель матрицы представлен на рисунке 15.

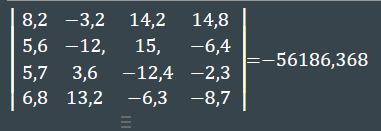


Рисунок – Проверка корня уравнения