**ОТЧЁТ**

**ПО ЛАБОРАТОРНОЙ РАБОТЕ № 2**

**ПРИБЛИЖЕННЫЕ МЕТОДЫ РЕШЕНИЯ**

**НЕЛИНЕЙНЫХ УРАВНЕНИЙ**

**(Вариант 9)**

*Выполнил студент 3 курса МОиАИС*

*Сагитов Александр*

***Постановка задачи:*** написать, отладить и выполнить программы решения систем линейных алгебраических уравнений, записанных в векторно-матричной форме Ax = b

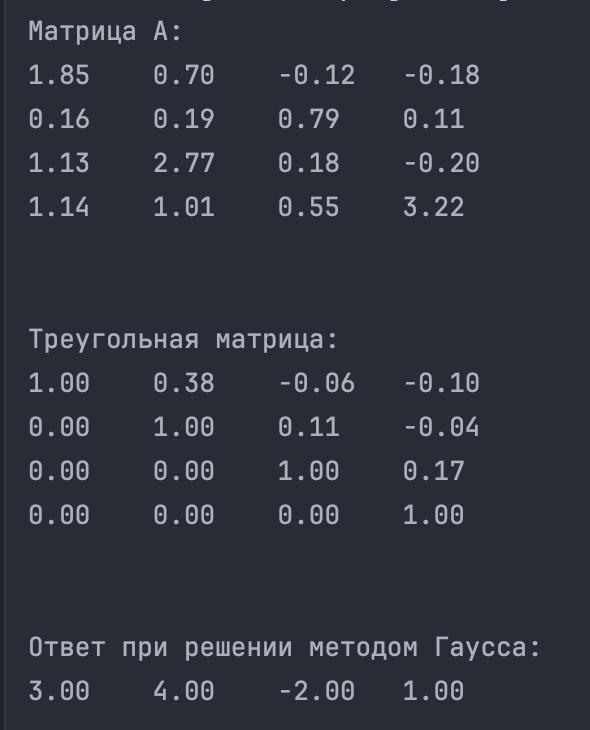
.

В х\* приведено точное решение.

Решить систему методом Гаусса с выбором главного элемента и методом Зейделя.

Оценить погрешности методов.

**Метод Гаусса**

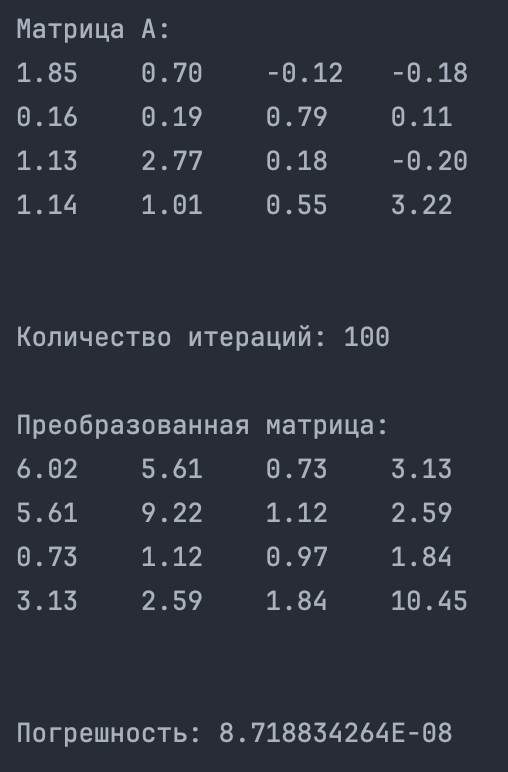


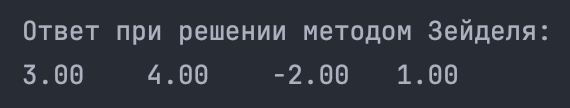
Погрешность:

Погрешность метода Гаусса равна 0, т.к. метод является точным. Однако в данной ситуации видна погрешность компьютерных вычислений.

**Метод Зейделя**

Решение СЛАУ методом Зейделя:





Погрешность:

Погрешность в методе Зейделя оценивается, как разность последней и предпоследней итерации.

**ПРИЛОЖЕНИЕ**

***Программа решения СЛАУ методом Гаусса***

import numpy as np  
  
N = 4  
  
  
def print\_matrix(matrix):  
 for row in matrix:  
 print("\t".join(f"{val:.2f}" for val in row))  
 print()  
  
  
def print\_vector(vector):  
 print("\t".join(f"{val:.2f}" for val in vector))  
 print()  
  
  
def swap\_rows(matrix, vector, row1, row2):  
 if row1 == row2:  
 return  
 matrix[[row1, row2]] = matrix[[row2, row1]]  
 vector[row1], vector[row2] = vector[row2], vector[row1]  
  
  
def gauss(A, b):  
 for i in range(N):  
 # Найдем строку с максимальным элементом по модулю в i-ом столбце  
 max\_row = i + np.argmax(np.abs(A[i:, i]))  
  
 # Перестановка строк, если нужно  
 if max\_row != i:  
 swap\_rows(A, b, i, max\_row)  
  
 # Приводим диагональный элемент к единице  
 leader = A[i, i]  
 A[i, :] /= leader  
 b[i] /= leader  
  
 # Зануляем элементы ниже главной диагонали  
 for j in range(i + 1, N):  
 factor = A[j, i]  
 A[j, i:] -= factor \* A[i, i:]  
 b[j] -= factor \* b[i]  
  
 # Обратный ход метода Гаусса для поиска решения  
 solution = np.zeros(N)  
 for i in range(N - 1, -1, -1):  
 sum\_ax = b[i] - np.dot(A[i, i + 1:], solution[i + 1:])  
 solution[i] = sum\_ax  
  
 print("\nТреугольная матрица:")  
 print\_matrix(A)  
 return solution  
  
  
# Пример использования  
A1 = np.array([[1.85, 0.70, -0.12, -0.18],  
 [0.16, 0.19, 0.79, 0.11],  
 [1.13, 2.77, 0.18, -0.20],  
 [1.14, 1.01, 0.55, 3.22]])  
  
b1 = np.array([8.41, -0.23, 13.91, 9.58])  
  
print("Матрица A:")  
print\_matrix(A1)  
  
x1 = gauss(A1.copy(), b1.copy())  
  
print("\nОтвет при решении методом Гаусса:")  
print\_vector(x1)

***Программа решения СЛАУ методом Зейделя***

import numpy as np  
  
N = 4  
  
  
def print\_matrix(matrix):  
 for row in matrix:  
 print("\t".join(f"{val:.2f}" for val in row))  
 print()  
  
  
def print\_vector(vector):  
 print("\t".join(f"{val:.2f}" for val in vector))  
 print()  
  
  
def is\_diagonally\_dominant(A):  
 """Проверяет, является ли матрица диагонально доминирующей."""  
 for i in range(N):  
 sum\_row = sum(abs(A[i, j]) for j in range(N) if j != i)  
 if abs(A[i, i]) <= sum\_row:  
 return False  
 return True  
  
  
def make\_diagonally\_dominant(A, b):  
 """Пытается сделать матрицу диагонально доминирующей путем перестановки строк."""  
 for i in range(N):  
 max\_index = i + np.argmax(np.abs(A[i:, i]))  
 if max\_index != i:  
 A[[i, max\_index]] = A[[max\_index, i]]  
 b[i], b[max\_index] = b[max\_index], b[i]  
  
  
def transpose(matrix):  
 return np.transpose(matrix)  
  
  
def zeidel(A, b, x, max\_iterations=100, eps=1e-7):  
 if not is\_diagonally\_dominant(A):  
 print("Матрица не является диагонально доминирующей, перестановка строк...")  
 make\_diagonally\_dominant(A, b)  
 if not is\_diagonally\_dominant(A):  
 print("Не удалось сделать матрицу диагонально доминирующей.")  
 return None  
  
 C = np.zeros((N, N))  
 d = np.zeros(N)  
 AT = transpose(A)  
  
 # Вычисляем матрицу C = AT \* A и вектор d = AT \* b  
 for i in range(N):  
 for j in range(N):  
 C[i, j] = sum(AT[i, k] \* A[k, j] for k in range(N))  
  
 for i in range(N):  
 d[i] = sum(AT[i, j] \* b[j] for j in range(N))  
  
 iterations = 0  
 max\_inaccuracy = 0  
  
 for \_ in range(max\_iterations):  
 x\_new = np.zeros(N)  
 for i in range(N):  
 x\_new[i] = d[i]  
 for j in range(N):  
 if j != i:  
 x\_new[i] -= C[i, j] \* x[j]  
 x\_new[i] /= C[i, i]  
  
 # Вычисляем максимальную погрешность  
 max\_inaccuracy = max(abs(x[i] - x\_new[i]) for i in range(N))  
  
 if max\_inaccuracy < eps:  
 break  
  
 x = np.copy(x\_new)  
 iterations += 1  
  
 print(f"\nКоличество итераций: {iterations}")  
 print("\nПреобразованная матрица:")  
 print\_matrix(C)  
 print(f"\nПогрешность: {max\_inaccuracy:.8f}")  
 return x  
  
# Пример использования  
A2 = np.array([[1.85, 0.70, -0.12, -0.18],  
 [0.16, 0.19, 0.79, 0.11],  
 [1.13, 2.77, 0.18, -0.20],  
 [1.14, 1.01, 0.55, 3.22]])  
  
  
x2 = np.zeros(N)  
b2 = np.array([8.41, -0.23, 13.91, 9.58])  
  
print("Матрица A:")  
print\_matrix(A2)  
  
x2 = zeidel(A2, b2, x2)  
  
if x2 is not None:  
 print("\nОтвет при решении методом Зейделя:")  
 print\_vector(x2)