Министерство образования Республики Беларусь
Учреждение образования
“БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ИНФОРМАТИКИ И РАДИОЭЛЕКТРОНИКИ”

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина: Методы численного анализа

ОТЧЁТ
к лабораторной работе 1
на тему

Решение систем линейных алгебраических уравнений (СЛАУ) методом Гаусса и с помощью его модификаций

Выполнил: студент группы 053502
Герчик Артём Вадимович

Проверил: Анисимов Владимир Яковлевич

Минск 2022

Оглавление

Цели выполнения задания 3
Краткие теоретические сведения 4
Задание 8
Программная реализация 9
Полученные результаты 18
Оценка 23
Выводы 24

Цели выполнения задания

1) Изучить метод Гаусса и его модификации, составить алгоритм метода и программу его реализации, получить численное решение заданной СЛАУ;
2) Составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
3) Составить программу решения СЛАУ по разработанному алгоритму;
4) Выполнить тестовые примеры и проверить правильность работы программы;

Краткие теоретические сведения

Задача отыскать решения СЛАУ с n неизвестными является одной из наиболее часто встречающихся вычислительных задач. Хотя задача решения СЛАУ сравнительно редко представляет самостоятельней интерес для приложений, от умения эффективно решать такие системы часто зависит сама возможность математического моделирования с применением ЭВМ разнообразных процессов. Значительная часть численных методов решения различных по своей природе задач (в особенности – нелинейных) включает в себя решение систем линейных уравнений как элементарный шаг соответствующего алгоритма.

СЛАУ обычно записывается в виде

$\sum_{j=1}^{n} a_j x_j = b_i; i \le 1 \le n$, или коротко Ax=b, где

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \cdots & \cdots & \cdots & \cdots \\ a_{n1} & a_{n2} & \cdots & a_{nm} \end{bmatrix}; \; a = \begin{bmatrix} x_1 \\ \cdots \\ x_n \end{bmatrix}; \; b = \begin{bmatrix} b_1 \\ \cdots \\ b_n \end{bmatrix}.$$

Здесь A и b заданы и требуется найти x.

Методы решения СЛАУ делятся на прямые и итерационные.

Прямые методы дают в принципе точное решение за конечное число арифметических операций. Они просты и наиболее универсальны. Для хорошо обусловленной системы необходимого порядка $n \le 200$ применяются практически только прямые методы.

Наибольшее распространение среди методов получил метод Гаусса и его модификации.

Метод Гаусса

Одним из самых распространенных методов решения систем линейных уравнений является метод Гаусса. Этот метод (который также называют *методом последовательного исключения неизвестных*) известен в различных вариантах.

Вычисления с помощью метода Гаусса заключаются в последовательном исключении неизвестных из системы для преобразования ее к эквивалентной системе с верхней треугольной матрицей. Вычисления значений неизвестных производят на этапе обратного хода.

1. Схема единственного деления.

Рассмотрим сначала простейший вариант метода Гаусса.

Прямой ход состоит из $n - 1$ шагов исключения.

1-й шаг. Целью этого шага является исключение неизвестного x_1 из уравнений с номерами $i = 2, 3, \ldots, n$. Предположим, что коэффициент $a_{11} \neq 0$. Будем называть его *главным элементом* 1-*го шага*.

Найдем величины

$qi_1 = ai_1/a_{11}$ $(i = 2, 3, \ldots, n)$,

называемые *множителями* 1-*го шага*. Вычтем последовательно из второго, третьего, …, n-го уравнений системы первое уравнение, умноженное соответственно на $q2_1, q_{31}, ..., qn_1$. Это позволит обратить в нуль коэффициенты при x_1 во всех уравнениях, кроме первого. В результате получим эквивалентную систему

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_1nxn = b_1$,

$a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_2n^{(1)}xn = b_2^{(1)}$,

$a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots + a_3n^{(1)}xn = b_3^{(1)}$,

. . . . . .

$an_2^{(1)}x_2 + an_3^{(1)}x_3 + \ldots + ann^{(1)}xn = bn^{(1)}$.

в которой $aij^{(1)}$ и $bij^{(1)}$ вычисляются по формулам

$aij^{(1)} = aij - qi_1a_1j$, $bi^{(1)} = bi - qi_1b_1$.

2-й шаг. Целью этого шага является исключение неизвестного x_2 из уравнений с номерами $i = 3, 4, \ldots, n$. Пусть $a_{22}^{(1)} \neq 0$, где $a_{22}^{(1)}$ – коэффициент, называемый *главным* (или *ведущим*) *элементом*2-*го шага*. Вычислим множители 2-го шага

$qi_2 = ai_2^{(1)} / a_{22}^{(1)}$ $(i = 3, 4, \ldots, n)$

и вычтем последовательно из третьего, четвертого, …, n-го уравнения системы второе уравнение, умноженное соответственно на $q_{32}, q_{42}, \ldots, qm_2$.

В результате получим систему:

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_1nxn = b_1$,

$a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_2n^{(1)} = b_2^{(1)}$,

$a_{33}^{(2)}x_3 + \ldots + a_3n^{(2)}xn = b_3^{(2)}$,

. . . . . .

$an_3^{(2)}x_3 + \ldots + ann^{(2)}xn = bn^{(2)}$.

Здесь коэффициенты $aij^{(2)}$ и $bij^{(2)}$ вычисляются по формулам

$aij^{(2)} = aij^{(1)} - qi_2a_2j^{(1)}$, $bi^{(2)} = bi^{(1)} - qi_2b_2^{(1)}$.

Аналогично проводятся остальные шаги. Опишем очередной k-й шаг.

k-й шаг. В предположении, что *главный* (*ведущий*) *элемент* k-го шага $akk^{(k-1)}$ отличен от нуля, вычислим *множители k-го шага*

$qik = aik^{(k-1)} / akk^{(k-1)}$ $(i = k + 1, \ldots, n)$

и вычтем последовательно из (k + 1)-го, …, n-го уравнений полученной на предыдущем шаге системы k-е уравнение, умноженное соответственно на $qk_{+1,}k$, $qk_{+2,}k$, …, qnk.

После (n - 1)-го шага исключения получим систему уравнений

$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_1nxn = b_1$,

$a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots + a_2n^{(1)}xn = b_2^{(1)}$,

$a_{33}^{(2)}x_3 + \ldots + a_3n^{(2)}xn = b_3^{(2)}$,

. . . . . .

$annn^{(n-1)}xn = bn^{(n-1)}$.

матрица $A^{(n-1)}$ которой является верхней треугольной. На этом вычисления прямого хода заканчиваются.

Обратный ход. Из последнего уравнения системы находим xn. Подставляя найденное значение xnв предпоследнее уравнение, получим xn_{-1}. Осуществляя обратную подстановку, далее последовательно находим xn_{-1}, xn_{-2}, …, x_1. Вычисления неизвестных здесь проводятся по формулам

$xn = bn^{(n-1)} / ann^{(n-1)}$,

$xk = (bn^{(k-1)} - ak,k_{+1}^{(k-1)}xk_{+1} - \ldots - akn^{(k-1)}xn) / akk^{(k-1)}$, ($k = n - 1, \ldots, 1$).

Необходимость выбора главных элементов. Заметим, что вычисление множителей, а также обратная подстановка требуют деления на главные элементы $akk^{(k-1)}$. Поэтому если один из главных элементов оказывается равным нулю, то схема единственного деления не может быть реализована. Здравый смысл подсказывает, что и в ситуации, когда все главные элементы отличны от нуля, но среди них есть близкие к нулю, возможен неконтролируемый рост погрешности.

2. Метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора).

Описание метода. На k-м шаге прямого хода коэффициенты уравнений системы с номерами $i = k+ 1, \ldots, n$ преобразуются по формулам

$aij^{(k)} = aij^{(k-1)} - qikakj$, $bi^{(k)} = bi^{(k-1)} - qikbk^{(k-1)}$, $i = k + 1, \ldots, n$.

Интуитивно ясно, что во избежание сильного роста коэффициентов системы и связанных с этим ошибок нельзя допускать появления больших множителей qik.

В методе Гаусса с выбором главного элементоа по столбцу гарантируется, что $|qik| \le 1$ для всех $k= 1, 2, \ldots, n - 1$ и $i = k + 1, \ldots, n$. Отличие этого варианта метода Гаусса от схемы единственного деления заключается в том, что на k-м шаге исключения в качестве главного элемента выбирают максимальный по модулю коэффициент $aikk$ при неизвестной xk в уравнениях с номерами $i = k +$

1, …, n. Затем соответствующее выбранному коэффициенту уравнение с номером ik меняют местами с k-м уравнением системы для того, чтобы главный элемент занял место коэффициента $akk^{(k-1)}$. После этой перестановки исключение неизвестного xk производят, как в схеме единственного деления.

3. Метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора).

В этой схеме допускается нарушение естественного порядка исключения неизвестных.

На 1-м шаге метода среди элементов aij определяют максимальный по модулю элемент ai_1j_1. Первое уравнение системы и уравнение с номером i_1 меняют местами. Далее стандартным образом производят исключение неизвестного xi_1 из всех уравнений, кроме первого.

На k-м шаге метода среди коэффициентов $aij^{(k-1)}$ при неизвестных в уравнениях системы с номерами $i = k, \ldots, n$ выбирают максимальный по модулю коэффициент $aikjk^{(k-1)}$. Затем k-е уравнение и уравнение, содержащее найденный коэффициент, меняют местами и исключают неизвестное xjk из уравнений с номерами $i = k + 1, \ldots, n$.

На этапе обратного хода неизвестные вычисляют в следующем порядке: $xjn, xjn_{-1}, \ldots, xj_1$.

Задание

Методом Гаусса и методом выбора главного элемента, найти с точностью 0.0001 численное решение системы **Ax=b**, где A = kC +D, A - исходная матрица для расчета, k - номер варианта (7), матрицы C, D и вектор свободных членов b задаются ниже.

Исходные данные:

Вектор **b**= $(4{,}2;\ 4{,}2;\ 4{,}2\ ;\ 4{,}2;\ 4{,}2)^{T}$,

$$C = \begin{bmatrix} 0{,}2 & 0 & 0{,}2 & 0 & 0 \\ 0 & 0{,}2 & 0 & 0{,}2 & 0 \\ 0{,}2 & 0 & 0{,}2 & 0 & 0{,}2 \\ 0 & 0{,}2 & 0 & 0{,}2 & 0 \\ 0 & 0 & 0{,}2 & 0 & 0{,}2 \end{bmatrix}, \quad D = \begin{bmatrix} 2{,}33 & 0{,}81 & 0{,}67 & 0{,}92 & -0{,}53 \\ -0{,}53 & 2{,}33 & 0{,}81 & 0{,}67 & 0{,}92 \\ 0{,}92 & -0{,}53 & 2{,}33 & 0{,}81 & 0{,}67 \\ 0{,}67 & 0{,}92 & -0{,}53 & 2{,}33 & 0{,}81 \\ 0{,}81 & 0{,}67 & 0{,}92 & -0{,}53 & 2{,}33 \end{bmatrix}.$$

Программная реализация

```
import numpy


def initial():
    print("\n")
    print("Решение систем линейных алгебраических
уравнений (СЛАУ) методом Гаусса и с помощью его
модификаций")
    print("Вариант 7 \n")


def inputValues():
    numpy.set_printoptions(suppress=True, precision=4,
floatmode="fixed")
    k = 7
    b = numpy.array([[4.2], [4.2], [4.2], [4.2], [4.2]])
    C = numpy.array([
        [0.2, 0.0, 0.2, 0.0, 0.0],
        [0.0, 0.2, 0.0, 0.2, 0.0],
        [0.2, 0.0, 0.2, 0.0, 0.2],
        [0.0, 0.2, 0.0, 0.2, 0.0],
        [0.0, 0.0, 0.2, 0.0, 0.2]
    ])
    D = numpy.array([
        [2.33, 0.81, 0.67, 0.92, -0.53],
        [-0.53, 2.33, 0.81, 0.67, 0.92],
        [0.92, -0.53, 2.33, 0.81, 0.67],
        [0.67, 0.92, -0.53, 2.33, 0.81],
        [0.81, 0.67, 0.92, -0.53, 2.33]
    ])
    A = k * C + D

    # # test example 1 normal
```

```
    # A = numpy.array([
    #     [2.0, 1.0],
    #     [1.0, -2.0]
    # ])
    # b = numpy.array([[3.0], [1.0]])
    # # test example 1 normal

    # # test example 2 normal
    # A = numpy.array([
    #     [4.0, 1.0, 1.0],
    #     [1.0, 4.0, 1.0],
    #     [1.0, 1.0, 4.0]
    # ])
    # b = numpy.array([[1.0], [1.0], [1.0]])
    # # test example 2 normal

    # # test example 3 with unlimited solutions but 0 on
main matrix isn't square
    # A = numpy.array([
    #     [1.0, 2.0, 1.0, 1.0, 3.0, 1.0],
    #     [1.0, 2.0, 1.0, 2.0, 1.0, -1.0],
    #     [1.0, 2.0, 1.0, -1.0, 5.0, -1.0],
    #     [1.0, 2.0, 1.0, -2.0, -4.0, 4.0]
    # ])
    # b = numpy.array([[7.0], [1.0], [2.0], [-1.0]])
    # # test example 3 with unlimited solutions but 0 on
main matrix isn't square

    # # test example 4 with unlimited solutions but 0 on
main matrix isn't square
    # A = numpy.array([
    #     [1.0, 2.0, -3.0, 5.0],
    #     [1.0, 3.0, -13.0, 22.0],
    #     [3.0, 5.0, 1.0, -2.0],
    #     [2.0, 3.0, 4.0, -7.0]
```

```
    # ])
    # b = numpy.array([[1.0], [-1.0], [5.0], [4.0]])
    # # test example 4 with unlimited solutions but 0 on
main matrix isn't square

    # # test example 5 no solutions but 0 on main matrix
isn't square
    # A = numpy.array([
    #     [2.0, -1.0, 3.0],
    #     [2.0, -1.0, -1.0],
    #     [4.0, -2.0, 6.0],
    #     [6.0, 8.0, -7.0]
    # ])
    # b = numpy.array([[1.0], [-2.0], [0.0], [2.0]])
    # # test example 5 no solutions but 0 on main matrix
isn't square

    # # test example 6
    # A = numpy.array([
    #     [3.0, 2.0, -5.0],
    #     [2.0, -1.0, 3.0],
    #     [1.0, 2.0, -1.0]
    # ])
    # b = numpy.array([[-1.0], [13.0], [9.0]])
    # # test example 6

    print("b:\n", b)
    print("\nC:\n", C)
    print("\nD:\n", D)
    print("\nA:\n", A)

    return A, b


def elementIsZero():
```

```
    print("Element == 0!")
    quit()


def checkIsUnlimitedOfSolutions(A, n):
    for i in range(n - 1, 0, -1):
        counter = 0
        for j in range(A.shape[1]):
            if numpy.around(A[i][j], decimals=4) != 0:
                counter += 1
        if counter == 1:
            return
        else:
            print('Бесконечное множество решений!')
            quit()


def checkIsNoSolutions(A, n, b):
    for i in range(n - 1, 0, -1):
        counter = 0
        for j in range(A.shape[1]):
            if numpy.around(A[i][j], decimals=4) != 0:
                counter += 1
        if counter == 0 and b[i] != 0:
            print('Нет решений!')
            quit()
        else:
            return


def baseMethod(A, b):
    if A.shape[0] != A.shape[1]:
        print("Матрица не квадратная!")
        quit()
    n = min(A.shape[0], A.shape[1])  # length of array
```

```
    for k in range(n):  # n is stop
        for i in range(k + 1, n):  # (start, stop, step)

            if A[k, k] == 0.0:
                # checkIsNoSolutions(A, n, b)
                # checkIsUnlimitedOfSolutions(A, n)
                elementIsZero()
            q = A[i, k] / A[k, k]

            for j in range(n):
                A[i, j] -= A[k, j] * q
            b[i] -= b[k] * q

    print("\nПреобразованный вектор b, схемой
единственного деления:\n", b)
    print("\nПреобразованная матрица A, схемой
единственного деления:\n", A)
    checkIsNoSolutions(A, n, b)
    checkIsUnlimitedOfSolutions(A, n)
    x = numpy.zeros((n, 1))

    for i in range(n - 1, -1, -1):  # (start, stop, step)
        for j in range(i + 1, n):
            b[i] -= A[i, j] * x[j]
        if A[i, i] == 0.0:
            elementIsZero()
        x[i] = b[i] / A[i, i]

    output("\nСхема единственного деления", x)


def partialChoiceMethod(A, b):
    if A.shape[0] != A.shape[1]:
        print("Матрица не квадратная!")
```

```
        quit()
    n = min(A.shape[0], A.shape[1])  # length of array

    for k in range(n):  # n is stop

        maxEl = k

        for temp in range(k, n):
            if abs(A[temp, k]) > abs(A[maxEl, k]):
                maxEl = temp
        A[[k, maxEl]] = A[[maxEl, k]]
        b[[k, maxEl]] = b[[maxEl, k]]

        for i in range(k + 1, n):  # (start, stop, step)

            if A[k, k] == 0.0:
                # checkIsUnlimitedOfSolutions(A, n)
                elementIsZero()
            q = A[i, k] / A[k, k]

            for j in range(n):
                A[i, j] -= A[k, j] * q
            b[i] -= b[k] * q

    print("\nПреобразованный вектор b, схемой частичного
выбора:\n", b)
    print("\nПреобразованная матрица A, схемой частичного
выбора:\n", A)
    checkIsNoSolutions(A, n, b)
    checkIsUnlimitedOfSolutions(A, n)
    x = numpy.zeros((n, 1))

    for i in range(n - 1, -1, -1):  # (start, stop, step)
        for j in range(i + 1, n):
            b[i] -= A[i, j] * x[j]
```

```
        if A[i, i] == 0.0:
            checkIsUnlimitedOfSolutions(A, n)
            elementIsZero()
        x[i] = b[i] / A[i, i]

    output("\nСхема частичного выбора ", x)


def fullChoiceMethod(A, b):
    if A.shape[0] != A.shape[1]:
        print("Матрица не квадратная!")
        quit()
    n = min(A.shape[0], A.shape[1])  # length of array

    for i in range(n):

        maxEl = (i, 0)

        for k in range(i, n):
            for j in range(n):
                if abs(A[k, j]) > abs(A[maxEl[0],
maxEl[1]]):
                    maxEl = (k, j)  # position of max
element
        A[[i, maxEl[0]]] = A[[maxEl[0], i]]  # swapping
rows
        b[[i, maxEl[0]]] = b[[maxEl[0], i]]

        for k in range(n):
            if k != i:  # if == - main element
                curElForQ02 = A[i, i]
                if curElForQ02 == 0.0:
                    # checkIsUnlimitedOfSolutions(A, n)
                    elementIsZero()
                curElForQ01 = A[k, i]
```

```
                q = curElForQ01 / curElForQ02
                for j in range(n):
                    A[k, j] -= A[i, j] * q
                b[k] -= b[i] * q

    print("\nПреобразованный вектор b, схемой полного
выбора:\n", b)
    print("\nПреобразованная матрица A, схемой полного
выбора:\n", A)
    checkIsNoSolutions(A, n, b)
    checkIsUnlimitedOfSolutions(A, n)
    x = numpy.zeros((n, 1))

    for j in range(n):

        maxEl = (j, 0)

        for i in range(n):
            if abs(A[i, j]) > abs(A[maxEl[0], maxEl[1]]):
                maxEl = (i, j)
        if A[maxEl[0], maxEl[1]] == 0.0:
            elementIsZero()
        x[j] = b[maxEl[0]] / A[maxEl[0], maxEl[1]]

    output("\nСхема полного выбора ", x)


def output(whichMethod, x):
    print(whichMethod)
    print("x:")
    print(x.T)

    numpy.set_printoptions(suppress=True, precision=10,
floatmode="fixed")
    print("x:")
```

```
    print(x.T)
    numpy.set_printoptions(suppress=True, precision=4,
floatmode="fixed")


def main():
    initial()
    AToWork, bToWork = inputValues()
    baseMethod(AToWork.copy(), bToWork.copy())
    partialChoiceMethod(AToWork.copy(), bToWork.copy())
    fullChoiceMethod(AToWork.copy(), bToWork.copy())

    # print('\n\n\n test')
    # matrix = numpy.array([
    #     [3.0, 2.0, -5.0],
    #     [0.0, -1.0, 3.0],
    #     [0.0, 2.0, -1.0]
    # ])
    # checkIsUnlimitedOfSolutions(matrix,
matrix.shape[0])
    # matrix = numpy.array([
    #     [3.0, 2.0, -5.0],
    #     [0.0, -1.0, 3.0],
    #     [0.0, 0.0, 0.0]
    # ])
    # b = numpy.array([[4.2], [4.2], [4.2]])
    # checkIsNoSolutions(matrix, matrix.shape[0], b)


if __name__ == '__main__':
    main()
```

Полученные результаты

Тестовый пример 1

Методом Гаусса найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 2x_1 + x_2 = 3 \\ x_1 - 2x_2 = 1 \end{cases}$$

Ответ:

Схема единственного деления	Схема частичного выбора	Схема полного выбора
$x_1 = 1.4000$ $x_2 = 0.2000$	$x_1 = 1.4000$ $x_2 = 0.2000$	$x_1 = 1.4000$ $x_2 = 0.2000$

Тестовый пример 2

Методом Гаусса найти с точностью 0,0001 численное решение СЛАУ:

$$\begin{cases} 4x_1 + x_2 + x_3 = 1 \\ x_1 + 4x_2 + x_3 = 1 \\ x_1 + x_2 + 4x_3 = 1 \end{cases}$$

Ответ:

Схема единственного деления	Схема частичного выбора	Схема полного выбора
$x_1 = 0.1667$ $x_2 = 0.1667$ $x_3 = 0.1667$	$x_1 = 0.1667$ $x_2 = 0.1667$ $x_3 = 0.1667$	$x_1 = 0.1667$ $x_2 = 0.1667$ $x_3 = 0.1667$

ЗАДАНИЕ

Вариант 7

Методом Гаусса найти с точностью 0,0001 численное решение системы **Ax=b**:

$$\begin{cases} 3.73x_1 + 0.81x_2 + 2.07x_3 + 0.92x_4 - 0.53x_5 = 4.2 \\ -0.53x_1 + 3.73x_2 + 0.81x_3 + 2.07x_4 + 0.92x_5 = 4.2 \\ 2.32x_1 - 0.53x_2 + 3.73x_3 + 0.81x_4 + 2.07x_5 = 4.2 \\ 0.67x_1 + 2.32x_2 - 0.53x_3 + 3.73x_4 + 0.81x_5 = 4.2 \\ 0.81x_1 + 0.67x_2 + 2.32x_3 - 0.53x_4 + 3.73x_5 = 4.2 \end{cases}$$

Ответ:

Схема единственного деления	Схема частичного выбора	Схема полного выбора
$x_1 = 0.8026$ $x_2 = 0.8032$ $x_3 = 0.2788$ $x_4 = 0.3727$ $x_5 = 0.6870$	$x_1 = 0.8026$ $x_2 = 0.8032$ $x_3 = 0.2788$ $x_4 = 0.3727$ $x_5 = 0.6870$	$x_1 = 0.8026$ $x_2 = 0.8032$ $x_3 = 0.2788$ $x_4 = 0.3727$ $x_5 = 0.6870$

Консоль:

```
/Users/lnxd/Desktop/MCHA/LAB1/venv/bin/python /Users/lnxd/Desktop/MCHA/LAB1/main.py


Решение систем линейных алгебраических уравнений (СЛАУ) методом Гаусса и с помощью его модификаций
Вариант 7

b:
 [[4.2000]
 [4.2000]
```

```
 [4.2000]
 [4.2000]
 [4.2000]]

C:
 [[0.2000 0.0000 0.2000 0.0000 0.0000]
 [0.0000 0.2000 0.0000 0.2000 0.0000]
 [0.2000 0.0000 0.2000 0.0000 0.2000]
 [0.0000 0.2000 0.0000 0.2000 0.0000]
 [0.0000 0.0000 0.2000 0.0000 0.2000]]

D:
 [[ 2.3300  0.8100  0.6700  0.9200 -0.5300]
 [-0.5300  2.3300  0.8100  0.6700  0.9200]
 [ 0.9200 -0.5300  2.3300  0.8100  0.6700]
 [ 0.6700  0.9200 -0.5300  2.3300  0.8100]
 [ 0.8100  0.6700  0.9200 -0.5300  2.3300]]

A:
 [[ 3.7300  0.8100  2.0700  0.9200 -0.5300]
 [-0.5300  3.7300  0.8100  2.0700  0.9200]
 [ 2.3200 -0.5300  3.7300  0.8100  2.0700]
 [ 0.6700  2.3200 -0.5300  3.7300  0.8100]
 [ 0.8100  0.6700  2.3200 -0.5300  3.7300]]
```

Преобразованный вектор b, схемой единственного деления:

```
 [[4.2000]
 [4.7968]
 [2.8773]
 [2.3360]
 [2.1455]]
```

Преобразованная матрица A, схемой единственного деления:

```
 [[ 3.7300  0.8100  2.0700  0.9200 -0.5300]
 [ 0.0000  3.8451  1.1041  2.2007  0.8447]
 [ 0.0000  0.0000  2.7394  0.8295  2.6268]
 [ 0.0000  0.0000  0.0000  2.7823  1.8910]
 [ 0.0000  0.0000  0.0000  0.0000  3.1229]]

Схема единственного деления
x:
[[0.8026 0.8032 0.2788 0.3727 0.6870]]
x:
[[0.8025664283 0.8032448774 0.2787722709 0.3726628377
0.6869991316]]

Преобразованный вектор b, схемой частичного выбора:
 [[4.2000]
 [4.7968]
 [2.8773]
 [2.3360]
 [2.1455]]

Преобразованная матрица A, схемой частичного выбора:
 [[ 3.7300  0.8100  2.0700  0.9200 -0.5300]
 [ 0.0000  3.8451  1.1041  2.2007  0.8447]
 [ 0.0000  0.0000  2.7394  0.8295  2.6268]
 [ 0.0000  0.0000  0.0000  2.7823  1.8910]
 [ 0.0000  0.0000  0.0000  0.0000  3.1229]]

Схема частичного выбора
x:
[[0.8026 0.8032 0.2788 0.3727 0.6870]]
x:
[[0.8025664283 0.8032448774 0.2787722709 0.3726628377
0.6869991316]]

Преобразованный вектор b, схемой полного выбора:
```

```
 [[ 2.9936]
 [ 3.0886]
 [ 0.4819]
 [ 0.5315]
 [-6.6332]]

Преобразованная матрица A, схемой полного выбора:
 [[ 3.7300  0.0000  0.0000  0.0000  0.0000]
 [ 0.0000  3.8451  0.0000  0.0000  0.0000]
 [ 0.0000 -0.0000  1.7286  0.0000  0.0000]
 [ 0.0000 -0.0000  0.0000  1.4261  0.0000]
 [ 0.0000 -0.0000  0.0000  0.0000 -9.6553]]

Схема полного выбора
x:
[[0.8026 0.8032 0.2788 0.3727 0.6870]]
x:
[[0.8025664283 0.8032448774 0.2787722709 0.3726628377
0.6869991316]]

Process finished with exit code 0
```

Оценка

Абсолютная погрешность:

$$\Delta\left(x_1\right) = \left|x - x_1\right| = 0.8026 - 0.8025664283 = 0.0000335717$$
$$\Delta\left(x_2\right) = \left|x - x_2\right| = 0.8032 - 0.8032448774 = 0.0000448774$$
$$\Delta\left(x_3\right) = \left|x - x_3\right| = 0.2788 - 0.2787722709 = 0.0000277291$$
$$\Delta\left(x_4\right) = \left|x - x_4\right| = 0.3727 - 0.3726628377 = 0.0000371623$$
$$\Delta\left(x_5\right) = \left|x - x_5\right| = 0.6870 - 0.6869991316 = 0.0000008684$$

Относительная погрешность:

$$\delta\left(x_1\right) = \frac{\Delta\left(x_1\right)}{\left|x_1\right|} = \frac{0.0000335717}{0.8026} = 0.000041828681784$$
$$\delta\left(x_2\right) = \frac{\Delta\left(x_2\right)}{\left|x_2\right|} = \frac{0.0000448774}{0.8032} = 0.000055873256972$$
$$\delta\left(x_3\right) = \frac{\Delta\left(x_3\right)}{|x_3|} = \frac{0.0000277291}{0.2788} = 0.000099458751793$$
$$\delta\left(x_4\right) = \frac{\Delta\left(x_4\right)}{|x_4|} = \frac{0.0000371623}{0.3727} = 0.000099711027636$$
$$\delta\left(x_5\right) = \frac{\Delta\left(x_5\right)}{|x_5|} = \frac{0.0000008684}{0.6870} = 0.000001264046579$$

Выводы

Таким образом, в ходе выполнения лабораторной работы мы изучили метод Гаусса и его модификации, составили алгоритм метода и программу его реализации, получили численное решение заданной СЛАУ, составили алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ, составили программу решения СЛАУ по разработанному алгоритму, выполнили тестовые примеры и проверили правильность работы программы.