Министерство образования Республики Беларусь
Учреждение образования «Белорусский государственный университет информатики и радиоэлектроники»

Факультет компьютерных систем и сетей
Кафедра информатики
Дисциплина «Методы численного анализа»

ОТЧЕТ

к лабораторной работе №1

на тему:

«РЕШЕНИЕ СИСТЕМ ЛИНЕЙНЫХ АЛГЕБРАИЧЕСКИХ УРАВННИЙ (СЛАУ) МЕТОДОМ ГАУССА И С ПОМОЩЬЮ ЕГО МОДИФИКАЦИЙ»

БГУИР 1-39 03 02

Выполнил студент группы 253504
Дмитрук Богдан Ярославович

(дата, подпись студента)

Проверил
Анисимов Владимир Яковлевич

(дата, подпись преподавателя)

Минск 2023

СОДЕРЖАНИЕ

1. ЦЕЛИ ВЫПОЛНЕНИЯ ЗАДАНИЯ

- Изучить метод Гаусса и его модификации, составить алгоритм метода и программу его реализации, получить численное решение заданной СЛАУ;
- составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
- составить программу решения СЛАУ по разработанному алгоритму;
- выполнить тестовые примеры и проверить правильность работы программы.

2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ

СЛАУ обычно записывается в виде

$$\sum_{j=1}^{n} a_{ij} x_j = b_i; i \le 1 \le n \quad , \text{ или коротко} \quad \boldsymbol{Ax} = \boldsymbol{b} \, , \qquad (1.1)$$

где

$$\boldsymbol{A} = \begin{bmatrix} a_{11} & a_{12} & \dots & a_{1n} \\ a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & \dots \\ a_{n1} & a_{n2} & \dots & a_{nn} \end{bmatrix} \quad ; \quad \boldsymbol{a} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_n \end{bmatrix} \quad ; \quad \boldsymbol{b} = \begin{bmatrix} b_1 \\ b_2 \\ \dots \\ b_n \end{bmatrix}.$$

Здесь $\boldsymbol{A}$ и $\boldsymbol{b}$ заданы, требуется найти $\boldsymbol{x}$.

Метод Гаусса прекрасно подходит для решения систем линейных алгебраических уравнений (СЛАУ). Он обладает рядом преимуществ по сравнению с другими методами:

- во-первых, нет необходимости предварительно исследовать систему уравнений на совместность;
- во-вторых, методом Гаусса можно решать не только СЛАУ, в которых число уравнений совпадает с количеством неизвестных переменных и основная матрица системы невырожденная, но и системы уравнений, в которых число уравнений не совпадает с количеством неизвестных переменных или определитель основной матрицы равен нулю;
- в-третьих, метод Гаусса приводит к результату при сравнительно небольшом количестве вычислительных операций.

Метод Гаусса включает в себя прямой (приведение расширенной матрицы к ступенчатому виду, то есть получение нулей под главной диагональю) и обратный (получение нулей над главной диагональю расширенной матрицы) ходы. Прямой ход и называется методом Гаусса, обратный - методом Гаусса-Жордана, который отличается от первого только последовательностью исключения переменных.

Метод Гаусса идеально подходит для решения систем содержащих больше трех линейных уравнений, для решения систем уравнений, которые не являются квадратными (чего не скажешь про метод Крамера и матричный метод). То есть метод Гаусса - наиболее универсальный

метод для нахождения решения любой системы линейных уравнений, он работает в случае, когда система имеет бесконечно много решений или.

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

$$q_{i1} = a_{i1}/a_{11} \quad (i = 2, 3, \ldots, n),$$

называемые *множителями* 1-*го шага*. Вычтем последовательно из второго, третьего, …, n-го уравнений системы первое уравнение, умноженное соответственно на q_{21}, q_{31}, …, q_{n1}. Это позволит обратить в нуль коэффициенты при x_1 во всех уравнениях, кроме первого. В результате получим эквивалентную систему

$$a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \ldots + a_{1n}x_n = b_1,$$
$$a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \ldots . + a_{2n}^{(1)}x_n = b_2^{(1)},$$
$$a_{32}^{(1)}x_2 + a_{33}^{(1)}x_3 + \ldots . + a_{3n}^{(1)}x_n = b_3^{(1)},$$
$$\ldots\ldots\ldots\ldots\ldots\ldots$$
$$a_{n2}^{(1)}x_2 + a_{n3}^{(1)}x_3 + \ldots . + a_{nn}^{(1)}x_n = b_n^{(1)},$$

в которой $a_{ij}^{(1)}$ и $b_{ij}^{(1)}$ вычисляются по формулам

$$a_{ij}^{(1)} = a_{ij} - q_{i1}a_{1j} \quad , \quad b_i^{(1)} = b_i - q_{i1}b_1.$$

2-й шаг. Целью этого шага является ислючение 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

$q_{i2} = a_{i2}^{(1)} / a_{22}^{(1)}$ (i = 3, 4, …, n)

и вычтем последовательно из третьего, четвертого, …, n-го уравнений системы второе уравнение, умноженное соответственно на q_{32}, q_{42}, …, q_{n2}. В результате получим систему

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1, \\ \quad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \dots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ \quad\quad a_{33}^{(2)}x_3 + \dots + a_{3n}^{(2)}x_n = b_3^{(2)}, \\ \dots\dots\dots\dots\dots\dots \\ \quad\quad a_{n3}^{(2)}x_3 + \dots + a_{nn}^{(2)}x_n = b_n^{(2)}. \end{array}$$

Здесь коэффициенты $a_{ij}^{(2)}$ и $b_j^{(2)}$ вычисляются по формулам

$$a_{ij}^{(2)} = a_{ij}^{(1)} - q_{i2}a_{2j}^{(1)}, \quad b_i^{(2)} = b_i^{(1)} - q_{i2}b_2^{(1)}.$$

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

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

$$q_{ik} = a_{ik}^{(k-1)} / a_{kk}^{(k-1)} \quad (i = k + 1, \dots, n)$$

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

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

$$\begin{array}{l} a_{11}x_1 + a_{12}x_2 + a_{13}x_3 + \dots + a_{1n}x_n = b_1, \\ \quad a_{22}^{(1)}x_2 + a_{23}^{(1)}x_3 + \dots + a_{2n}^{(1)}x_n = b_2^{(1)}, \\ \quad\quad a_{33}^{(2)}x_3 + \dots + a_{3n}^{(2)}x_n = b_3^{(2)}, \\ \dots\dots\dots\dots\dots\dots \\ \quad\quad\quad a_{nn}^{(n-1)}x_n = b_n^{(n-1)}, \end{array}$$

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

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

$$x_n = b_n^{(n-1)} / a_{nn}^{(n-1)},$$

$$x_k = (b_n^{(k-1)} - a_{k,k+1}^{(k-1)} x_{k+1} - \ldots - a_{kn}^{(k-1)} x_n) / a_{kk}^{(k-1)}, (k = n-1, \ldots, 1).$$

Необходимость отличия от 0 главных элементов. Заметим, что вычисление множителей, а также обратная подстановка требуют деления на главные элементы $a_{kk}^{(k-1)}$. Поэтому если один из главных элементов оказывается равным нулю, то схема 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.

Метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора). На k-м шаге прямого хода коэффициенты уравнений системы с номерами $i = k + 1, \ldots, n$ преобразуются по формулам

$$a_{ij}^{(k)} = a_{ij}^{(k-1)} - q_{ik} a_{kj}, \; b_i^{(k)} = b_i^{(k-1)} - q_{ik} b_k^{(k-1)}, \; i = k+1, \ldots, n.$$

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

В методе Гаусса с выбором главного элемента по столбцу гарантируется, что $|q_{ik}| \le 1$ 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 $a_{i_k k}$ при неизвестной x_k в уравнениях с номерами $i = k + 1, \ldots, n$. Затем соответствующее выбранному коэффициенту уравнение с номером i_k меняют местами с k-м уравнением системы для того, чтобы главный элемент занял место коэффициента $a_{kk}^{(k-1)}$. После этой перестановки исключение неизвестного x_k производят, как в схеме единственного деления.

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 a_{ij} определяют максимальный по модулю элемент a_{ij}. Первое уравнение системы и уравнение с номером i_i меняют местами. Далее стандартным образом производят исключение неизвестного x_i из всех уравнений, кроме первого.

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

На этапе обратного хода неизвестные вычисляют в следующем порядке: x_{jn} **,** x_{jn-1} **, …,** x_{j1} **.**

3. ЗАДАНИЕ

Методом Гаусса и методом выбора главного элемента найти с точностью 0,0001 численное решение системы ***Ax = b,*** где A = k*C + D, A – исходная матрица для расчёта, k – номер варианта (0-15), матрицы C, D и вектор свободных членов b задаются ниже.

Исходные данные:

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

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

4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ

```
#------------главный исполняемый файл-----------

import initial_data as init
import gaussian_elimination_classic as gec
import gaussian_elimination_col_mod as gecm
import gaussian_elimination_full_matrix_mod as gefm
import numpy as np

#M = init.test1()

#M = init.test2()

M = init.test3()
M = gec.forward_move(M, (lambda x, y, z: None))
c_solution = gec.backward_move(M)

print(np.round(c_solution.astype(float), 4))

M = init.get_initial_matrix()
M = gec.forward_move(M, (lambda x, y, z: None))
c_solution = gec.backward_move(M)

print(np.round(c_solution.astype(float), 4))

M = init.get_initial_matrix()
M = gec.forward_move(M, gecm.col_mod)
c_solution = gec.backward_move(M)


print(np.round(c_solution.astype(float), 4))


M = init.get_initial_matrix()
arr = [i for i in range(M.shape[1] - 1)]
M = gec.forward_move(M, gefm.full_mod, arr)
c_solution = gec.backward_move(M, arr)

print(np.round(c_solution.astype(float), 4))

#------------файл инициализации матрицы----------------

from fractions import Fraction as fr
```

```
import numpy as np

C = np.array([
        [fr(0.2), 0, fr(0.2), 0, 0],
        [0, fr(0.2), 0, fr(0.2), 0],
        [fr(0.2), 0, fr(0.2), 0, fr(0.2)],
        [0, fr(0.2), 0, fr(0.2), 0],
        [fr(0.2), 0, fr(0.2), 0, fr(0.2)]
    ])

D = np.array([
        [fr(2.33), fr(0.81), fr(0.67), fr(0.92), fr(-0.53)],
        [fr(-0.53), fr(2.33), fr(0.81), fr(0.67), fr(0.92)],
        [fr(0.92), fr(-0.53), fr(2.33), fr(0.81), fr(0.67)],
        [fr(0.67), fr(0.92), fr(-0.53), fr(2.33), fr(0.81)],
        [fr(0.81), fr(0.67), fr(0.92), fr(-0.53), fr(2.33)]
    ])

b = np.array([
        [fr(4.2)],
        [fr(4.2)],
        [fr(4.2)],
        [fr(4.2)],
        [fr(4.2)]
    ])

k = 5


def test1():
    M = np.array([
        [fr(2),fr(3)],
        [fr(4),fr(6)]
    ])

    B = np.array([
        [fr(7)],
        [fr(14)]
    ])

    col = M.shape[0]
    M = np.transpose(M)
    for c in range(col):
        if all(i == 0 for i in M[c]):
```

```
            M = np.delete(M, c, axis = 0)
    M = np.transpose(M)

    M_ex = np.hstack((M,B))

    if np.linalg.matrix_rank(M.astype(float)) !=
np.linalg.matrix_rank(M_ex.astype(float)):
        print("the matrix is inconsistent")
        exit(0)
    if np.linalg.matrix_rank(M.astype(float)) != len(M[0]):
        print("the matrix has an infinite number of
solutions")
        exit(0)
    return M_ex



def test2():
    M = np.array([
        [2,3],
        [4,6]
    ])
    B = np.array([
        [7],
        [12]
    ])
    col = M.shape[0]
    M = np.transpose(M)
    for c in range(col):
        if all(i == 0 for i in M[c]):
            M = np.delete(M, c, axis = 0)
    M = np.transpose(M)
    M_ex = np.hstack((M,B))
    if np.linalg.matrix_rank(M.astype(float)) !=
np.linalg.matrix_rank(M_ex.astype(float)):
        print("the matrix is inconsistent")
        exit(0)
    if np.linalg.matrix_rank(M.astype(float)) != len(M[0]):
        print("the matrix has an infinite number of
solutions")
        exit(0)
    return M_ex

def test3():
```

```
    M = np.array([
        [2.,3.],
        [4.,-5.]
    ])
    B = np.array([
        [7.],
        [-6.]
    ])
    col = M.shape[0]
    M = np.transpose(M)
    for c in range(col):
        if all(i == 0 for i in M[c]):
            M = np.delete(M, c, axis = 0)
    M = np.transpose(M)

    M_ex = np.hstack((M,B))

    if np.linalg.matrix_rank(M.astype(float)) !=
np.linalg.matrix_rank(M_ex.astype(float)):
        print("the matrix is inconsistent")
        exit(0)
    if np.linalg.matrix_rank(M.astype(float)) != len(M[0]):
        print("the matrix has an infinite number of
solutions")
        exit(0)
    return M_ex

def get_initial_matrix():
    M = k*C + D

    col = M.shape[0]
    M = np.transpose(M)
    for c in range(col):
        if all(i == 0 for i in M[c]):
            M = np.delete(M, c, axis = 0)
    M = np.transpose(M)

    M_ex = np.hstack((M,b))
    if np.linalg.matrix_rank(M.astype(float)) !=
np.linalg.matrix_rank(M_ex.astype(float)):
        print("the matrix is inconsistent")
        exit(0)
    if np.linalg.matrix_rank(M.astype(float)) != len(M[0]):
```

```
        print("the matrix has an infinite number of
solutions")
        exit(0)
    return M_ex

#файл реализации метода Гаусса с передаваемой в него функцией
модификации

from fractions import Fraction as fr
import numpy as np

def forward_move(M, mod_func, arr = None):
    row, col = M.shape

    for cur_col in range(col - 1):
        mod_func(M, cur_col, arr)

        if M[cur_col][cur_col] == 0:
            for r in range(cur_col, row):
                if M[r][cur_col] != 0:
                    M[r], M[cur_col] = M[cur_col].copy(),
M[r].copy()
                    break
        deviser = M[cur_col][cur_col]
        for r in range(cur_col + 1, row):
            M[r] -= M[cur_col]*M[r][cur_col] / deviser
    return M

def backward_move(M, arr = None):
    row, col = M.shape
    solutions = np.zeros(col, dtype=fr)
    for r in range(row - 1, -1, -1):
        obtained = np.dot(M[r, r + 1:], solutions[r + 1:])
        a = fr(M[r][col - 1] - obtained)
        solutions[r] = a/M[r][r]

    if arr:
        resort(solutions, arr)
    return solutions[:col - 1]



def resort(solutions, arr):
    for i in range(len(solutions) - 1):
```

```
        swap = arr.index(i)
        a, b, c, d = solutions[swap], solutions[i], arr[swap],
arr[i]
        solutions[i], solutions[swap], arr[i], arr[swap] = a,
b, c, d

#------------файл модификации выбора по столбцу--------------

from fractions import Fraction as fr
import numpy as np
def col_mod(M, cur_col, arr):
    pivot = np.argmax(np.abs(M[cur_col:, cur_col])) + cur_col
    M[pivot], M[cur_col] = M[cur_col].copy(), M[pivot].copy()

#-----------файл модификации полного выбора------------------
from fractions import Fraction as fr
import numpy as np
import copy

def full_mod(M, cur_col, arr = None):
    row, col = M.shape
    mr, mc = cur_col, cur_col

    max = np.abs(M[cur_col][cur_col])
    for r in range(cur_col, row):
        for c in range(cur_col, col - 1):
            if np.abs(M[r][c]) > max:
                max = np.abs(M[r][c])
                mr, mc = r,c

    arr[cur_col], arr[mc] = arr[mc], arr[cur_col]

    M[mr], M[cur_col] = M[cur_col].copy(), M[mr].copy()

    M[:, [cur_col, mc]] = M[:, [mc, cur_col]]
```

5. ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ

Тестовая матрица 1:

```
[[ 2.  3.  7.]
 [ 4.  6. 14.]]
```

Вывод программного продукта:

```
the matrix has an infinite number of solutions
```

Тестовая матрица 2:

```
[[ 2.  3.  7.]
 [ 4.  6. 12.]]
```

Вывод программного продукта:

```
the matrix is inconsistent
```

Тестовая матрица 3:

```
[[ 2.  3.  7.]
 [ 4. -5. -6.]]
```

Вывод программного продукта:

```
[0.7727 1.8182]
```

Матрица варианта 5, составленная в соответствии с условием

```
[[ 3.33  0.81  1.67  0.92 -0.53  4.2 ]
 [-0.53  3.33  0.81  1.67  0.92  4.2 ]
 [ 1.92 -0.53  3.33  0.81  1.67  4.2 ]
 [ 0.67  1.92 -0.53  3.33  0.81  4.2 ]
 [ 1.81  0.67  1.92 -0.53  3.33  4.2 ]]
```

Значения переменных, полученные после использования классического метода Гаусса:

```
[0.6877 0.7651 0.5748 0.6502 0.5056]
```

Значения переменных, полученные после использования схемы частичного выбора:

```
[0.6877 0.7651 0.5748 0.6502 0.5056]
```

Значения переменных, полученные после использования схемы полного выбора:

```
[0.6877 0.7651 0.5748 0.6502 0.5056]
```

6. ОЦЕНКА

Подсчет абсолютной погрешности приближенного решения:

$$\Delta(x_1) = |x - x_1| = 0.6877 - 0.68771426 = 0.00001426$$

$$\Delta(x_2) = |x - x_2| = 0.7651 - 0.76512113 = 0.00002113$$

$$\Delta(x_3) = |x - x_3| = 0.5748 - 0.57479370 = 0.00000630$$

$$\Delta(x_4) = |x - x_4| = 0.6502 - 0.65024257 = 0.00004257$$

$$\Delta(x_5) = |x - x_5| = 0.5056 - 0.50559480 = 0.00000520$$

Относительная погрешность приближенного числа:

$$\delta(x_1) = \frac{\Delta(x_1)}{|x_1|} = \frac{0.0.00001426}{0.6877} = 0.0000207357859532$$

$$\delta(x_2) = \frac{\Delta(x_2)}{|x_2|} = \frac{0.00002113}{0.7651} = 0.0000276173049275$$

$$\delta(x_3) = \frac{\Delta(x_3)}{|x_3|} = \frac{0.00000630}{0.5748} = 0.0000109603340292$$

$$\delta(x_4) = \frac{\Delta(x_4)}{|x_4|} = \frac{0.00004257}{0.6502} = 0.0000654721624116$$

$$\delta(x_5) = \frac{\Delta(x_5)}{|x_5|} = \frac{0.00000520}{0.5056} = 0.0000102848101266$$

7. ВЫВОДЫ

Таким образом, в ходе выполнения данной лабораторной работы был применён метод Гаусса, метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора) и метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора) для решения системы линейных уравнений, составлены алгоритмы и созданы реализации соответствующих программ на языке Python для решения поставленной задачи, также проведена оценка.

Итак, метод Гаусса применим к любой системе линейных уравнений, он идеально подходит для решения систем, содержащих больше трех линейных уравнений. Метод Гаусса решения СЛАУ с числовыми коэффициентами в силу простоты и однотипности выполняемых операций пригоден для счета на электронно-вычислительных машинах.

Достоинства метода:

1. Менее трудоёмкий по сравнению с другими методами;
2. Позволяет однозначно установить, совместна система или нет, и если совместна, найти её решение;
3. Позволяет найти максимальное число линейно независимых уравнений – ранг матрицы системы.