ФЕДЕРАЛЬНОЕ ГОСУДАРСТВЕННОЕ БЮДЖЕТНОЕ
ОБРАЗОВАТЕЛЬНОЕ УЧРЕЖДЕНИЕ ВЫСШЕГО ОБРАЗОВАНИЯ
«СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ
ТЕЛЕКОММУНИКАЦИЙ И ИНФОРМАТИКИ»

КАФЕДРА ПМиК

ОТЧЁТ ПО ЛАБОРАТОРНОЙ РАБОТЕ №1

«Решение систем линейных уравнений методом Жордана-Гаусса»
по дисциплине «Алгоритмы и вычислительные методы оптимизации»

Выполнил: студент гр. ИП-814
Краснов И.В.
Проверил: ассистент кафедры ПМиК
Новожилов Д. И.

Новосибирск 2021

Содержание

Задание

Решение систем линейных уравнений методом Жордана-Гаусса

Написать программу, находящую решение системы линейных уравнений методом Жордана-Гаусса.

Программа должна выводить промежуточные матрицы после каждого шага исключений и решение системы. Программа должна работать для различных тестов: система имеет единственное решение, система имеет бесконечно много решений, система не имеет решения.

Должна иметься возможность быстро ввести входные данные для различного количества переменных и уравнений. Начальную работу программу необходимо продемонстрировать на предложенной ниже системе (система выбирается по номеру бригады).

Текст программы

```
import sys
'''import copy
import itertools'''

def gcd(m, n):
    while m % n != 0:
        oldm = m
        oldn = n

        m = oldn
        n = oldm % oldn
    return n

def DeleteLine(matrix, rows, cols, line):
    if line == rows:
        rows -= 1
    else:
        for i in range(line + 1, rows):
            for j in range(cols):
                matrix[i][j], matrix[i - 1][j] = matrix[i - 1][j], matrix[i][j]
        rows -= 1
    return rows

def PrintMatrix(matrix, rows, cols):
    print()
    for i in range(rows):
        for j in range(cols):
            if j == (cols - 1):
                print("|%20s" %matrix[i][j], end='')
            else:
                print("%20s" %matrix[i][j], end='')
        print()

def gauss(matrix, rows, cols):
    k = 0
    for t in range(rows):
        if matrix[k][k].num == 0:
            k += 1
            continue
        for i in range(rows):
            if k == i:
                continue
            for j in range(k + 1, cols):
                matrix[i][j] = ((matrix[k][k] * matrix[i][j]) - (matrix[i][k] *
matrix[k][j])) / matrix[k][k]
        for i in range(rows):
            if k == i:
                continue
            matrix[i][k] = Fraction()
        k += 1
        print(t, ") ")
        PrintMatrix(matrix, rows, cols)
        print()
    k = 0
    for i in range(rows):
        if matrix[k][k].num == 0:
            k += 1
            continue
        for j in range(cols - 1, k - 1, -1):
```

```
            matrix[i][j] /= matrix[k][k]
        k += 1
    i = 0
    while i < rows:
        zero_line = True
        for k in range(cols - 1):
            if matrix[i][k].num != 0:
                zero_line = False
                break
        if zero_line and matrix[i][cols - 1].num != 0:
            PrintMatrix(matrix, rows, cols)
            print("Нет решения!")
            sys.exit()
        elif zero_line == True and matrix[i][cols - 1].num == 0:
            rows = DeleteLine(matrix, rows, cols, i)
        i += 1
        '''if (zero_line):
            continue'''
    PrintMatrix(matrix, rows, cols)
    return rows


class Fraction:
    def __init__(self, top=0, bottom=1):
        if bottom != 0:
            self.num = top
            self.den = bottom
            if self.den < 0:
                self.num *= -1
                self.den *= -1
            tmp = gcd(self.num, self.den)
            self.num = self.num // tmp
            self.den = self.den // tmp
        else:
            print("Denominator == 0")
            sys.exit()

    def __str__(self):
        if self.den == 1:
            return str(self.num)
        else:
            return str(self.num) + "/" + str(self.den)

    def show(self):
        if self.den == 1:
            print(self.num)
        else:
            print(self.num, "/", self.den)

    def __add__(self, otherfraction):
        return Fraction(self.num * otherfraction.den + self.den * otherfraction.num,
self.den * otherfraction.den)

    def __iadd__(self, other):
        tmp = Fraction(self.num * other.den + self.den * other.num, self.den *
other.den)
        self = tmp
        return tmp

    def __isub__(self, other):
        tmp = Fraction(self.num * other.den - self.den * other.num, self.den *
```

```
other.den)
        self = tmp
        return tmp

    def __imul__(self, other):
        tmp = Fraction(self.num * other.num, self.den * other.den)
        self = tmp
        return tmp

    def __itruediv__(self, other):
        tmp = Fraction(self.num * other.den, self.den * other.num)
        self = tmp
        return tmp

    def __sub__(self, otherfraction):
        return Fraction(self.num * otherfraction.den - self.den * otherfraction.num,
self.den * otherfraction.den)

    def __mul__(self, otherfraction):
        return Fraction(self.num * otherfraction.num, self.den * otherfraction.den)

    def __truediv__(self, otherfraction):
        return Fraction(self.num * otherfraction.den, self.den * otherfraction.num)


    def __eq__(self, other):
        firstnum = self.num * other.den
        secondnum = other.num * self.den
        return firstnum == secondnum

    def __ne__(self, other):
        firstnum = self.num * other.den
        secondnum = other.num * self.den
        return firstnum != secondnum


number_of_rows = int(input())
number_of_cols = int(input())
matrixCin = [[0] * number_of_cols for i in range(number_of_rows)]
matrix = [[Fraction(0, 1)] * number_of_cols for i in range(number_of_rows)]

for i in range(number_of_rows):
    stri = input()
    matrixCin[i] = stri.strip().split(" ")
    for j in range(number_of_cols):
        matrix[i][j] = Fraction(int(matrixCin[i][j]), 1)

PrintMatrix(matrix, number_of_rows, number_of_cols)

number_of_rows = gauss(matrix, number_of_rows, number_of_cols)
sign = Fraction(-1, 1)
for i in range(number_of_rows):
    print('x', i + 1, " = ", matrix[i][number_of_cols - 1], end=" ")
    for j in range(number_of_cols - 1):
        if j != i and matrix[i][j].num != 0:
            if matrix[i][j].num > 0:
                print(" - ", matrix[i][j], 'x', j + 1, end=" ")
            else:
                print(" + ", matrix[i][j] * sign, 'x', j + 1, end=" ")
    print()
```

Результат работы программы

```
15              -5              8               11              -6|             -76
15               1              7                1              11|             -79
-5              11              5               -9              10|              -6
13              -5             -1               11               3|             -27
15               4             -3               -1               3|              -4
```

Рис. 1 Система из задания

```
                1               0               0               0               0|              -2
                0               1               0               0               0|               3
                0               0               1               0               0|              -6
                0               0               0               1               0|               1
                0               0               0               0               1|              -1
x 1  =  -2
x 2  =  3
x 3  =  -6
x 4  =  1
x 5  =  -1
```

Рис. 2 Решение системы

```
                1               2               3|              5
                2               4               6|              6
0 )

                1               2               3|              5
                0               0               0|             -4


                1               2               3|              5
                0               0               0|             -4
Нет решения!
```

Рис. 3 Система, которая не имеет решения

```
 4              -3              -2               1|             -2
 3              -1              -2               0|              1
 2               1              -2              -1|              4
```

```
                1               0           -4/5            -1/5|           1
                0               1           -2/5            -3/5|           2
x 1  =  1  +  4/5 x 3  +  1/5 x 4
x 2  =  2  +  2/5 x 3  +  3/5 x 4
```

Рис. 4, 5 Система, имеющая бесконечно много решений