Федеральное государственное бюджетное образовательное учреждение высшего образования «Сибирский государственный университет телекоммуникаций и

информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №2

по дисциплине «Алгоритмы и вычислительные методы оптимизации»

Бригада №2

Выполнили:

студенты группы ИП-014

Обухов А. И.

Гулая А. С.

Малышев В. А.

Работу проверил: Новожилов Д.И.

Новосибирск 2023 г.

Задания

Написать программу, находящую все базисные решения системы линейных уравнений методом Жордана-Гаусса. Программа должна выводить промежуточные матрицы после каждого шага исключений и все найденные базисные 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). Для получения максимальной оценки необходимо, чтобы все вычисления выполнялись в простых дробях. Для этого использовать класс простых дробей, реализованный в лабораторной 1.

Прикрепите архив с файлами отчета и исходных кодов программы лабораторной 2. Отчет должен содержать: титульный лист, задание, текст программы, результаты тестирования. В отчете должен быть тест, предложенный в задании соответствующего бригаде варианта. Отчет присылается для каждого члена бригады от его имени. Каждую лабораторную необходимо защитить, только после этого будут выставлены баллы.

Листинг

import math

import itertools

class Fraction:

\_\_slots\_\_ = ('\_numerator', '\_denominator')

def \_\_init\_\_(self, numerator=0, denominator=1):

if type(numerator) is not int or type(denominator) is not int:

raise TypeError(

'Fraction(%s, %s) - the numerator and denominator values must be integers'

% (numerator, denominator))

if denominator == 0:

raise ZeroDivisionError('Fraction(%s, 0)' % numerator)

g = math.gcd(numerator, denominator)

if denominator < 0:

g = -g

numerator //= g

denominator //= g

self.\_numerator = numerator

self.\_denominator = denominator

def from\_str(fraction):

fraction = fraction.split('/')

tmp\_numerator = int(fraction[0])

tmp\_denominator = 1 if len(fraction) == 1 else int(fraction[1])

return Fraction(tmp\_numerator, tmp\_denominator)

def \_\_add\_\_(self, other):

"""x + y"""

if isinstance(other, Fraction):

return Fraction(

self.\_numerator \* other.\_denominator +

other.\_numerator \* self.\_denominator,

self.\_denominator \* other.\_denominator)

return NotImplemented

def \_\_sub\_\_(self, other):

""" x - y"""

if isinstance(other, Fraction):

return Fraction(

self.\_numerator \* other.\_denominator -

other.\_numerator \* self.\_denominator,

self.\_denominator \* other.\_denominator)

return NotImplemented

def \_\_mul\_\_(self, other):

"""x \* y"""

if isinstance(other, Fraction):

return Fraction(self.\_numerator \* other.\_numerator,

self.\_denominator \* other.\_denominator)

return NotImplemented

def \_\_truediv\_\_(self, other):

"""x / y"""

if isinstance(other, Fraction):

return Fraction(self.\_numerator \* other.\_denominator,

self.\_denominator \* other.\_numerator)

return NotImplemented

def \_\_lt\_\_(self, other):

"""x < y"""

if isinstance(other, Fraction):

return self.\_numerator \* other.\_denominator < other.\_numerator \* self.\_denominator

return NotImplemented

def \_\_le\_\_(self, other):

"""x <= y"""

if isinstance(other, Fraction):

return self.\_numerator \* other.\_denominator <= other.\_numerator \* self.\_denominator

return NotImplemented

def \_\_eq\_\_(self, other):

"""x == y"""

if isinstance(other, Fraction):

return self.\_numerator \* other.\_denominator == other.\_numerator \* self.\_denominator

return NotImplemented

def \_\_ne\_\_(self, other):

"""x != y"""

if isinstance(other, Fraction):

return self.\_numerator \* other.\_denominator != other.\_numerator \* self.\_denominator

return NotImplemented

def \_\_gt\_\_(self, other):

"""x > y"""

if isinstance(other, Fraction):

return self.\_numerator \* other.\_denominator > other.\_numerator \* self.\_denominator

return NotImplemented

def \_\_ge\_\_(self, other):

"""x >= y"""

if isinstance(other, Fraction):

return self.\_numerator \* other.\_denominator >= other.\_numerator \* self.\_denominator

return NotImplemented

def \_\_repr\_\_(self):

if self.\_denominator == 1:

return 'Fraction(%s)' % self.\_numerator

else:

return 'Fraction(%s, %s)' % (self.\_numerator, self.\_denominator)

def \_\_str\_\_(self):

if self.\_denominator == 1:

return str(self.\_numerator)

else:

return '%s/%s' % (self.\_numerator, self.\_denominator)

def abs(self):

return Fraction(abs(self.\_numerator), abs(self.\_denominator))

def print\_matrix(matrix):

for i in matrix:

for j in i:

print(j, end=' ')

print()

print()

def read\_matrix\_from\_file(path):

matrix = []

f = open(path, 'r')

for line in f:

a = list(line.strip().split(' '))

for i in range(len(a)):

a[i] = Fraction.from\_str(a[i])

matrix.append(a)

return matrix

def calculate(matrix, flag=True):

shift = 0

for c in range(len(matrix)):

index = c

for i in range(c + 1, len(matrix)):

if matrix[index][c + shift].abs() <= matrix[i][c + shift].abs():

index = i

if index != c:

matrix[index], matrix[c] = matrix[c], matrix[index]

if matrix[c][c + shift] == Fraction(0):

shift += 1

zero\_counter = 0

for item in matrix[c]:

if item == Fraction(0):

zero\_counter += 1

if zero\_counter == len(matrix[c]):

continue

if matrix[c][c + shift] != Fraction(1):

matrix[c] = [i / matrix[c][c + shift] for i in matrix[c]]

print\_matrix(matrix)

for i in range(len(matrix)):

if matrix[i][c + shift] == Fraction(0):

continue

if i == c:

continue

coeff = matrix[i][c + shift] \* Fraction(-1)

for j in range(c, len(matrix[0])):

matrix[i][j] = matrix[i][j] + matrix[c][j] \* coeff

print\_matrix(matrix)

count\_no\_null\_str = 0

for i in matrix:

null\_sum\_flag = True

for j in i[:-1]:

if j != Fraction(0):

null\_sum\_flag = False

break

if null\_sum\_flag and i[-1] != Fraction(0):

count\_no\_null\_str = 0

break

elif null\_sum\_flag and i[-1] == Fraction(0):

continue

count\_no\_null\_str += 1

if not count\_no\_null\_str:

return None

elif count\_no\_null\_str == len(matrix) and count\_no\_null\_str == len(

matrix[0]) - 1:

return [[matrix[i][-1] for i in range(len(matrix))]]

else:

res = [[], []]

# Общее решение

for i in matrix:

tmp\_sum = Fraction(0)

for j in i:

tmp\_sum += j.abs()

if tmp\_sum != Fraction(0):

res[0].append(i)

matrix = res[0]

if flag:

for i in itertools.combinations([i for i in range(len(matrix[0]) - 1)],

count\_no\_null\_str):

print("-" \* 30)

print(i)

tmp\_res = [0 for i in range(len(matrix[0]) - 1)]

tmp\_matrix = []

# Получение необходимых столбов, как строк

for j in i:

tmp\_matrix.append([x[j] for x in matrix])

tmp\_matrix.append([x[-1] for x in matrix])

# Транспонирование

tmp\_matrix = [list(j) for j in zip(\*tmp\_matrix)]

res\_r = calculate(tmp\_matrix, False)

if not res\_r:

print("\nНет решения")

elif len(res\_r) == 1:

print("\nРешение:")

ans = res\_r[0]

for j in range(len(ans)):

print("x{} = {}".format(j + 1, ans[j]), end=" ")

t = 0

for j in i:

tmp\_res[j] = ans[t]

t += 1

print("\n")

res[1].append(tmp\_res)

elif len(res\_r) == 2:

print("\nСЛАУ имеет множество решений")

return res

if \_\_name\_\_ == '\_\_main\_\_':

matrix = read\_matrix\_from\_file('input.txt')

res = calculate(matrix)

print("#" \* 50)

print("<Сокращенная матрица>")

print\_matrix(matrix)

if not res:

print("\nНет решения")

elif len(res) == 1:

print("\nРешение:")

ans = res[0]

for i in range(len(ans)):

print("\tx{} = {}".format(i + 1, ans[i]))

print("\n")

elif len(res) == 2:

o\_ans = res[0]

ans = res[1]

print("\nОбщее решение:")

for i in o\_ans:

tmp\_str = ""

for j in range(len(i) - 1):

tmp = i[j]

if tmp != Fraction(0):

if tmp.abs() != Fraction(1):

if tmp < Fraction(0):

tmp\_str += " - "

else:

tmp\_str += " + "

tmp\_str += str(tmp.abs()) + " \* "

tmp\_str += "x" + str(j + 1)

tmp\_str += " = " + str(i[-1])

print(tmp\_str)

print("\nБазисные решения:")

for i in ans:

for j in range(len(i)):

print("\tx{} = {}".format(j + 1, i[j]))

print()

else:

print("Вышла какая-то ошибочка :с")

Результаты работы программы

input.txt

1 0 1 1 3 7 1

1 1 0 0 5 3 1

1 0 0 0 4 1 1

output

1 0 0 0 4 1 1

0 1 0 0 1 2 0

0 0 1 1 -1 6 0

1 0 0 0 4 1 1

0 1 0 0 1 2 0

0 0 1 1 -1 6 0

1 0 0 0 4 1 1

0 1 0 0 1 2 0

0 0 1 1 -1 6 0

------------------------------

(0, 1, 2)

1 0 0 1

0 1 0 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 1, 3)

1 0 0 1

0 1 0 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 1, 4)

1 0 4 1

0 1 1 0

0 0 -1 0

1 0 4 1

0 1 1 0

0 0 -1 0

1 0 4 1

0 1 1 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 1, 5)

1 0 1 1

0 1 2 0

0 0 6 0

1 0 1 1

0 1 2 0

0 0 6 0

1 0 1 1

0 1 2 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 2, 3)

1 0 0 1

0 0 0 0

0 1 1 0

1 0 0 1

0 1 1 0

0 0 0 0

СЛАУ имеет множество решений

------------------------------

(0, 2, 4)

1 0 4 1

0 0 1 0

0 1 -1 0

1 0 4 1

0 1 -1 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 2, 5)

1 0 1 1

0 0 2 0

0 1 6 0

1 0 1 1

0 1 6 0

0 0 2 0

1 0 1 1

0 1 6 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 3, 4)

1 0 4 1

0 0 1 0

0 1 -1 0

1 0 4 1

0 1 -1 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 3, 5)

1 0 1 1

0 0 2 0

0 1 6 0

1 0 1 1

0 1 6 0

0 0 2 0

1 0 1 1

0 1 6 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(0, 4, 5)

1 4 1 1

0 1 2 0

0 -1 6 0

1 4 1 1

0 1 -6 0

0 1 2 0

1 0 25 1

0 1 -6 0

0 0 8 0

1 0 25 1

0 1 -6 0

0 0 1 0

1 0 0 1

0 1 0 0

0 0 1 0

Решение:

x1 = 1 x2 = 0 x3 = 0

------------------------------

(1, 2, 3)

1 0 0 0

0 0 0 1

0 1 1 0

1 0 0 0

0 1 1 0

0 0 0 1

1 0 0 0

0 1 1 0

0 0 0 1

Нет решения

------------------------------

(1, 2, 4)

1 0 1 0

0 0 4 1

0 1 -1 0

1 0 1 0

0 1 -1 0

0 0 4 1

1 0 1 0

0 1 -1 0

0 0 1 1/4

1 0 0 -1/4

0 1 0 1/4

0 0 1 1/4

Решение:

x1 = -1/4 x2 = 1/4 x3 = 1/4

------------------------------

(1, 2, 5)

1 0 2 0

0 0 1 1

0 1 6 0

1 0 2 0

0 1 6 0

0 0 1 1

1 0 0 -2

0 1 0 -6

0 0 1 1

Решение:

x1 = -2 x2 = -6 x3 = 1

------------------------------

(1, 3, 4)

1 0 1 0

0 0 4 1

0 1 -1 0

1 0 1 0

0 1 -1 0

0 0 4 1

1 0 1 0

0 1 -1 0

0 0 1 1/4

1 0 0 -1/4

0 1 0 1/4

0 0 1 1/4

Решение:

x1 = -1/4 x2 = 1/4 x3 = 1/4

------------------------------

(1, 3, 5)

1 0 2 0

0 0 1 1

0 1 6 0

1 0 2 0

0 1 6 0

0 0 1 1

1 0 0 -2

0 1 0 -6

0 0 1 1

Решение:

x1 = -2 x2 = -6 x3 = 1

------------------------------

(1, 4, 5)

1 1 2 0

0 4 1 1

0 -1 6 0

1 1 2 0

0 1 1/4 1/4

0 -1 6 0

1 0 7/4 -1/4

0 1 1/4 1/4

0 0 25/4 1/4

1 0 7/4 -1/4

0 1 1/4 1/4

0 0 1 1/25

1 0 0 -8/25

0 1 0 6/25

0 0 1 1/25

Решение:

x1 = -8/25 x2 = 6/25 x3 = 1/25

------------------------------

(2, 3, 4)

1 1 -1 0

0 0 1 0

0 0 4 1

1 1 -1 0

0 0 1 1/4

0 0 1 0

1 1 0 1/4

0 0 1 1/4

0 0 0 -1/4

1 1 0 1/4

0 0 1 1/4

0 0 0 1

1 1 0 0

0 0 1 0

0 0 0 1

Нет решения

------------------------------

(2, 3, 5)

1 1 6 0

0 0 2 0

0 0 1 1

1 1 0 -6

0 0 1 1

0 0 0 -2

1 1 0 -6

0 0 1 1

0 0 0 1

1 1 0 0

0 0 1 0

0 0 0 1

Нет решения

------------------------------

(2, 4, 5)

1 -1 6 0

0 1 2 0

0 4 1 1

1 -1 6 0

0 1 1/4 1/4

0 1 2 0

1 0 25/4 1/4

0 1 1/4 1/4

0 0 7/4 -1/4

1 0 25/4 1/4

0 1 1/4 1/4

0 0 1 -1/7

1 0 0 8/7

0 1 0 2/7

0 0 1 -1/7

Решение:

x1 = 8/7 x2 = 2/7 x3 = -1/7

------------------------------

(3, 4, 5)

1 -1 6 0

0 1 2 0

0 4 1 1

1 -1 6 0

0 1 1/4 1/4

0 1 2 0

1 0 25/4 1/4

0 1 1/4 1/4

0 0 7/4 -1/4

1 0 25/4 1/4

0 1 1/4 1/4

0 0 1 -1/7

1 0 0 8/7

0 1 0 2/7

0 0 1 -1/7

Решение:

x1 = 8/7 x2 = 2/7 x3 = -1/7

##################################################

<Сокращенная матрица>

1 0 0 0 4 1 1

0 1 0 0 1 2 0

0 0 1 1 -1 6 0

Общее решение:

x1 + 4 \* x5x6 = 1

x2x5 + 2 \* x6 = 0

x3x4x5 + 6 \* x6 = 0

Базисные решения:

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 1

x2 = 0

x3 = 0

x4 = 0

x5 = 0

x6 = 0

x1 = 0

x2 = -1/4

x3 = 1/4

x4 = 0

x5 = 1/4

x6 = 0

x1 = 0

x2 = -2

x3 = -6

x4 = 0

x5 = 0

x6 = 1

x1 = 0

x2 = -1/4

x3 = 0

x4 = 1/4

x5 = 1/4

x6 = 0

x1 = 0

x2 = -2

x3 = 0

x4 = -6

x5 = 0

x6 = 1

x1 = 0

x2 = -8/25

x3 = 0

x4 = 0

x5 = 6/25

x6 = 1/25

x1 = 0

x2 = 0

x3 = 8/7

x4 = 0

x5 = 2/7

x6 = -1/7

x1 = 0

x2 = 0

x3 = 0

x4 = 8/7

x5 = 2/7

x6 = -1/7