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

информатики» (СибГУТИ)

Кафедра прикладной математики и кибернетики

Лабораторная работа №1

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 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). Для получения максимальной оценки необходимо, чтобы все вычисления выполнялись в простых дробях. Для этого реализовать класс простых дробей. Реализованный класс можно будет использовать в лабораторной №2 и курсовой работе

В отчете должно быть 3 теста: единственное решение, бесконечно много решений, нет решения. В качестве одного из тестов использовать систему в соответствующем варианте задания лабораторной. Отчет присылается для каждого члена бригады от его имени. Баллы выставляются только после защиты лабораторной. При несвоевременной защите баллы снижаются.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).

Для получения максимальной оценки необходимо, чтобы все вычисления

выполнялись в простых дробях. Для этого реализовать класс простых дробей.

Листинг

import math

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 get\_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):

shift = 0

for c in range(len(matrix)):

index = c

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

if matrix[index][c + shift].get\_abs() <= matrix[i][c + shift].get\_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 + shift] = [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)

no\_null\_str = 0

for i in matrix:

flag = True

for j in i[:-1]:

if j != Fraction(0):

flag = False

break

if flag and i[-1] != Fraction(0):

no\_null\_str = 0

break

elif flag and i[-1] == Fraction(0):

continue

no\_null\_str += 1

print("Ответ:")

if not no\_null\_str:

print("нет решения")

elif no\_null\_str == len(matrix) and no\_null\_str == len(matrix[0]) - 1:

tmp = list()

for i in range(len(matrix)):

tmp.append("x" + str(i + 1) + " = " + str(matrix[i][-1]))

print(\*tmp, sep='\n')

else:

tmp = list()

for i in range(len(matrix)):

str\_tmp = ""

if matrix[i][i] == Fraction(1):

str\_tmp += "x" + str(i + 1) + " = " + str(matrix[i][-1])

for j in range(len(matrix[0]) - 1):

if j == i:

continue

if matrix[i][j] == Fraction(0):

continue

str\_tmp += " + "

str\_tmp += "(" + str(matrix[i][j] \* Fraction(-1)) + ")" + "x" + str(j + 1)

else:

flag = True

for j in matrix[i][:-1]:

if j != Fraction(0):

flag = False

break

if flag and matrix[i][-1] == Fraction(0):

continue

for j in range(len(matrix[0]) - 1):

if matrix[i][j] == Fraction(0):

continue

str\_tmp += " + "

str\_tmp += "(" + str(matrix[i][j] \* Fraction(-1)) + ")" + "x" + str(j + 1)

str\_tmp += " = " + str(matrix[i][-1])

if str\_tmp != "":

tmp.append(str\_tmp)

print(\*tmp, sep='\n')

def main():

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

calculate(matrix)

return

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

main()

Результаты работы программы

Input.txt

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

output

1 4/15 -1/5 -1/15 1/5 -4/15

15 1 7 1 11 -79

-5 11 5 -9 10 -6

13 -5 -1 11 3 -27

15 -5 8 11 -6 -76

1 4/15 -1/5 -1/15 1/5 -4/15

0 -3 10 2 8 -75

0 37/3 4 -28/3 11 -22/3

0 -127/15 8/5 178/15 2/5 -353/15

0 -9 11 12 -9 -72

1 4/15 -1/5 -1/15 1/5 -4/15

0 1 12/37 -28/37 33/37 -22/37

0 -3 10 2 8 -75

0 -127/15 8/5 178/15 2/5 -353/15

0 -9 11 12 -9 -72

1 0 -53/185 5/37 -7/185 -4/37

0 1 12/37 -28/37 33/37 -22/37

0 0 406/37 -10/37 395/37 -2841/37

0 0 804/185 202/37 1471/185 -1057/37

0 0 515/37 192/37 -36/37 -2862/37

1 0 -53/185 5/37 -7/185 -4/37

0 1 12/37 -28/37 33/37 -22/37

0 0 1 192/515 -36/515 -2862/515

0 0 804/185 202/37 1471/185 -1057/37

0 0 406/37 -10/37 395/37 -2841/37

1 0 0 623/2575 -149/2575 -4378/2575

0 1 0 -452/515 471/515 622/515

0 0 1 192/515 -36/515 -2862/515

0 0 0 9886/2575 21257/2575 -11371/2575

0 0 0 -2246/515 5893/515 -8139/515

1 0 0 623/2575 -149/2575 -4378/2575

0 1 0 -452/515 471/515 622/515

0 0 1 192/515 -36/515 -2862/515

0 0 0 1 -5893/2246 8139/2246

0 0 0 9886/2575 21257/2575 -11371/2575

1 0 0 0 6479/11230 -28939/11230

0 1 0 0 -1559/1123 4928/1123

0 0 1 0 1020/1123 -7758/1123

0 0 0 1 -5893/2246 8139/2246

0 0 0 0 102914/5615 -102914/5615

1 0 0 0 6479/11230 -28939/11230

0 1 0 0 -1559/1123 4928/1123

0 0 1 0 1020/1123 -7758/1123

0 0 0 1 -5893/2246 8139/2246

0 0 0 0 1 -1

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

Ответ:

x1 = -2

x2 = 3

x3 = -6

x4 = 1

x5 = -1