«Министерство образования и науки Российской Федерации Федеральное государственное бюджетное образовательное учреждение высшего образования «Алтайский государственный технический университет им. И.И. Ползунова»

Факультет информационных технологий

Кафедра прикладной математики

Отчёт защищён с оценкой\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Преподаватель Проскурин А.В.

«\_\_\_\_\_»\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_2022 г.

Отчёт

Лабораторной работе №1

«Решить систему линейных уравнений методом Гаусса»

Студент группы ПИ 92 В.М. Шульпов

Преподаватель доцент, к. м. н. Проскурин А.В.

Барнаул 2022

**Задание**

Составить программу для решения системы линейных алгебраических уравнений методом Гаусса с выбором главного элемента, нахождения определителя матрицы системы и вычисления обратной матриц. Исходные данные – матрица системы уравнений и столбец свободных членов должны читаться из файла, а результаты расчетов помещаться в файл. В случае, когда матрица системы вырождена, выдать об этом сообщение. В противном случае вывести решение системы, невязки, величину определителя, обратную матрицу. Подобрать тестовые примеры, предусматриваемые различные ситуации (матрица вырожденная, невырожденная) и провести вычисления.

Теория:

*# Решение уравнений методом Гаусса  
#  
# P.S.  
# В функциях используются переменные a и b  
# a - матрица коэффициентов при неизвестных  
# b - столбец свободных коэффициентов  
#  
# ----- ----- ----- ----- ----- ----- ----- ----- ----- ----- ТЕОРИЯ:  
# --- Единичная матрица (E) - матрица, у которой на главной диагонали единицы, а на всех остальных местах нули.  
#  
# 1 0 0 Пример  
# 0 1 0  
# 0 0 1  
#  
# --- Умножение матрицы на её обратную матрицы даёт единичную матрицу: A \* A^(-1) = E или A^(-1) \* A = E  
# --- Чтобы найти определитель матрицы, нужно:  
# - Пример для матрицы 2x2:  
#  
# 1 2  
# 3 4  
#  
# определитель A = delta A = det A = 1\*2 - 3\*2 = -4  
#  
# - Пример для матрицы 3x3 (сводится к 2x2):  
#  
# 1 2 3  
# 4 5 6  
# 7 8 9  
#  
# det A = + 1 \* (5\*9 - 8\*6) - 2 \* (4\*9 - 7\*6) + 3 \* (4\*8-7\*5)  
#  
# Знаки:  
# + - +  
# - + -  
# + - +  
#  
# --- Вырожденная матрица - матрица, определитель которой равен НУЛЮ. У неё не может быть обратной матрицы.  
# --- Алгебраическое дополнение:  
#  
# 1 2 3  
# 4 5 6  
# 7 8 9  
# 1 \* 3  
# 4 \* 6  
# \* \* \*  
#  
# Алгебраическое дополнение A32 =  
# (-1)^(3+2) \* |1 3| = -1 \* (6-4\*3) = 6  
# |4 6|  
#  
# --- Обратная матрица  
#  
# A =  
# 1 2  
# 3 4  
#  
# A^(1) =  
# |A11 \* detA A21 \* detA|  
# |A12 \* detA A22 \* detA|  
#  
# --- Решение с выбором главного ведущего  
# Среди элементов a(k)sk, s=k,k+1,…,m находят наибольший по модулю,  
# который называют главным или ведущим элементом, и перестановкой строк выводят его на главную диагональ,  
# после чего выполняют цикл исключения.  
# Такая модификация алгоритма называется методом Гаусcа с выбором главного элемента.  
#  
# --- Величина невязки  
# Контроль вычислений можно вести по величине невязки - векторе,  
# который получается при вычитании из правой части системы левой, в которую подставлено полученное решение.  
#  
# --- Ортогона́льная ма́трица  
# - это квадратная матрица A с вещественными элементами,  
# результат умножения которой на транспонированную матрицу A^T равен единичной матрице[1]:  
# или, что эквивалентно, её обратная матрица (которая обязательно существует) равна транспонированной матрице  
#  
# --- Прямой ход метода Гаусса  
# заключается в приведении матрицы системы к треугольному виду  
#  
# --- Количество операций для решения  
# Для решений системы m линейных алгебраических уравнений с m неизвестными требуется порядка M алгебраических операций.  
# M равно: 2/3 \* m^3  
#*

В рамках реализации данного задания был создан двумерный массив, для хранения расширенной матрицы. Чтение расширенной матрицы происходит из файла input.dat построчно. Вся логика программы реализуется в функциях:

**def** count(func):  
 *"""декоратор - счётчик"""*  
  
**def** read\_input\_data(filename, matrix):  
 *"""чтение матрицы из файла"""*  
**def** write\_output\_matrix(filename, matrix, title=**""**):  
 *"""запись результатов в файл"""*  
  
**def** write\_output\_list(filename, \_list, title=**""**):  
 *"""запись результатов в файл"""*  
  
**def** format\_print(n, a, b=**None**, selected=**None**):  
 *"""вывод системы на экран"""*   
  
  
**def** check\_discrepancy(a, b, x):  
 *"""проверка на соотвествие (невязка)"""*  
  
@count  
**def** swap\_rows(a, b, row1, row2):  
 *"""перемена местами двух строк системы"""*  
  
**def** divide\_row(a, b, changeable\_row, divider):  
 *"""деление строки системы на число"""*  
  
  
**def** combine\_rows(a, b, changeable\_row, source\_row, multiplier):  
 *"""сложение строки системы с другой строкой, умноженной на число"""*  
  
  
**def** fill\_identity\_matrix(max\_i, max\_j):  
  
  
**def** solve\_by\_gauss(a, b, n):  
 *"""решение системы методом Гаусса (приведением к треугольному виду)"""*  
  
**def** transpose\_matrix(a, n):  
 *"""транспонировать матрицу"""*  
  
  
**def** get\_inverse\_matrix(a, n):  
 *"""получить обратную матрицу"""*  
  
  
**def** detailed\_solve\_by\_gauss(a, b, n):  
 *"""решение системы методом Гаусса (приведением к треугольному виду)"""*  
  
  
**def** calc\_det\_triangular\_matrix(matrix):  
 *"""получение определителя"""*

**Описание алгоритма:**

Изначально происходит заполнение расширенной матрицы данными из файла. Далее копирование содержимого матрицы в двумерный массив для дальнейшего поиска нормы матрицы. Затем в глобальном цикле по столбцам матрицы происходит поиск главного элемента каждого столбца, перестановка строки с этим элементом на главную диагональ и сразу же зануление элементов ниже главной диагонали, в цикле: пусть самый большой элемент - первый в текущем столбце, запоминание его номера и значения, если он равен нулю, установка соответствующего флага. Если в текущем столбце найден элемент больше первого, и он не нулевой - запоминание его значения и номера. Затем, если самый большой элемент найден, и он не нулевой – перестановка строк местами, если остались только нулевые элементы, в том числе на главной диагонали – вывод сообщения о том, что матрица вырождена. В этом же цикле, происходит поиск коэффициента умножения каждого элемента строки, чтобы занулить нижние элементы: элемент столбца, который необходимо занулить делиться на элемент главной диагонали текущего столбца и затем полученное значение для экономии памяти запоминается на месте элемента, которые надо было занулить, а вся текущая строка поэлементно умножается на данные коэффициент, и затем полученное значение вычитается из элементов текущей строки, результат запоминается в данном элементе строки. С помощью декоратора происходит подсчет кол-ва перестановок. Если нечетное, то определитель отрицательный. Таким образом, будет получено значение определителя, если матрица невырождена. Корни находятся в следующем цикле методом обратной прогонки: сначала вычисляется последний корень, затем предпоследний и так далее до первого корня. Затем происходит вывод корней, определителя. Далее вычисляется невязка для каждого корня: внешний цикл – прогонка по строкам, начиная с последней, обнуление значения невязки для каждого корня. Вложенный цикл – прогонка по столбцам в текущей строке: каждый элемент текущей строки домножается на соответствующ корень и значения складываются. Невязка = полученной значение во вложенном цикле минус значение, соответствующей строки матрицы b, полученной после зануления элементов. Обратная матрица находится путем решения n систем линейных уравнений, из единичной матрицы, поочередно подставляя каждый столбец данной матрицы на место последнего столбца расширенной матрицы.

**Текст программы:**

**import** csv  
INPUT\_FILENAME = **'input.csv'**OUTPUT\_FILENAME = **'output.csv'  
  
  
def** count(func):  
 *"""декоратор - счётчик"""* **def** wrapper(\*a, \*\*kw):  
 wrapper.count += 1  
 **return** func(\*a, \*\*kw)  
  
 wrapper.count = 0  
 **return** wrapper  
  
  
**def** read\_input\_data(filename, matrix):  
 *"""чтение матрицы из файла"""* **with** open(filename) **as** File:  
 reader = csv.reader(File, delimiter=**','**)  
 **for** row\_index, row **in** enumerate(reader):  
 matrix.append([])  
 **for** item **in** row:  
 matrix[row\_index].append(float(item))  
 print(**'Чтение из файла проведено успешно'**)  
  
  
**def** write\_output\_matrix(filename, matrix, title=**""**):  
 *"""запись результатов в файл"""* **with** open(filename, mode=**"a"**, encoding=**'utf-8'**) **as** FILE:  
 file\_writer = csv.writer(FILE, delimiter=**","**, lineterminator=**"\r"**)  
 file\_writer.writerow([title])  
 **for** row **in** matrix:  
 file\_writer.writerow(row)  
 print(**f'\nЗапись в файл проведена успешно ({**title**})'**)  
  
  
**def** write\_output\_list(filename, \_list, title=**""**):  
 *"""запись результатов в файл"""* **with** open(filename, mode=**"a"**, encoding=**'utf-8'**) **as** FILE:  
 file\_writer = csv.writer(FILE, delimiter=**","**, lineterminator=**"\r"**)  
 file\_writer.writerow([title])  
 file\_writer.writerow([\_list])  
 print(**f'\nЗапись в файл проведена успешно ({**title**})'**)  
  
  
**def** format\_print(n, a, b=**None**, selected=**None**):  
 *"""вывод системы на экран"""* **for** row\_ind **in** range(n):  
 print(**"("**, end=**''**)  
 **for** col\_ind **in** range(n):  
 selection\_label = (**"" if** selected **is None or** selected != (row\_ind, col\_ind) **else "<"**)  
 print(**"\t{1:10.2f}{0}"**.format(selection\_label, a[row\_ind][col\_ind]), end=**''**)  
 **if** b **is not None**:  
 print(**"\t) = (\t{1:10.2f})"**.format(row\_ind + 1, b[row\_ind]))  
 **else**:  
 print(**"\t)"**)  
  
  
**def** check\_discrepancy(a, b, x):  
 *"""проверка на соотвествие (невязка)"""* print(**"\nПодсчёт невязки:"**)  
 text\_discrepancy = **''  
 for** row\_ind **in** range(len(b)):  
 \_sum = 0  
 **for** col\_ind **in** range(len(a[row\_ind])):  
 \_sum += a[row\_ind][col\_ind] \* x[col\_ind]  
 text\_discrepancy += **f'Невязка в {**row\_ind + 1**}-й строке = {**\_sum - b[row\_ind]**}\n'** print(**f'Невязка в {**row\_ind + 1**}-й строке = {**\_sum - b[row\_ind]**}\n'**)  
 **return** text\_discrepancy  
  
  
@count  
**def** swap\_rows(a, b, row1, row2):  
 *"""перемена местами двух строк системы"""* a[row1], a[row2] = a[row2], a[row1]  
 b[row1], b[row2] = b[row2], b[row1]  
  
  
**def** divide\_row(a, b, changeable\_row, divider):  
 *"""деление строки системы на число"""* a[changeable\_row] = [a / divider **for** a **in** a[changeable\_row]]  
 b[changeable\_row] /= divider  
  
  
**def** combine\_rows(a, b, changeable\_row, source\_row, multiplier):  
 *"""сложение строки системы с другой строкой, умноженной на число"""* a[changeable\_row] = [(a + k \* multiplier) **for** a, k **in** zip(a[changeable\_row], a[source\_row])]  
 b[changeable\_row] += b[source\_row] \* multiplier  
  
  
**def** fill\_identity\_matrix(max\_i, max\_j):  
 identity\_matrix = [[0 **for** j **in** range(max\_j)] **for** i **in** range(max\_i)]  
 **for** i **in** range(0, max\_i):  
 **for** j **in** range(0, max\_j):  
 **if** i == j:  
 identity\_matrix[i][j] = 1  
 **else**:  
 identity\_matrix[i][j] = 0  
 **return** identity\_matrix  
  
  
**def** solve\_by\_gauss(a, b, n):  
 *"""решение системы методом Гаусса (приведением к треугольному виду)"""* column = 0  
 **while** column < n:  
 current\_row = **None  
 for** r **in** range(column, n):  
 **if** current\_row **is None or** abs(a[r][column]) > abs(a[current\_row][column]):  
 current\_row = r  
 **if** current\_row **is None**:  
 **return None  
 if** current\_row != column:  
 swap\_rows(a, b, current\_row, column)  
 **try**:  
 divide\_row(a, b, column, a[column][column])  
 **except** ZeroDivisionError:  
 **raise** ZeroDivisionError  
 **for** r **in** range(column + 1, n):  
 combine\_rows(a, b, r, column, -a[r][column])  
 column += 1  
 det = calc\_det\_triangular\_matrix(a)  
 **if** det == 0:  
 **return None  
 else**:  
 x = [0 **for** \_ **in** b]  
 **for** i **in** range(n - 1, -1, -1):  
 x[i] = b[i] - sum(x \* a **for** x, a **in** zip(x[(i + 1):], a[i][(i + 1):])) *# zip создает итератор кортежей* **return** x  
  
  
**def** transpose\_matrix(a, n):  
 *"""транспонировать матрицу"""* transposed\_matrix = [[0 **for** j **in** range(n)] **for** i **in** range(n)]  
 **for** i **in** range(n):  
 **for** j **in** range(n):  
 transposed\_matrix[j][i] = a[i][j]  
 **return** transposed\_matrix  
  
  
**def** get\_inverse\_matrix(a, n):  
 *"""получить обратную матрицу"""* inverse\_matrix = [] *# обартная матрица* identity\_row = [0 **for** j **in** range(n)] *# столбец свободных слагаемых* **for** i **in** range(n):  
 **for** j **in** range(n):  
 **if** i == j:  
 identity\_row[j] = 1  
 **else**:  
 identity\_row[j] = 0  
 inverse\_matrix\_col = solve\_by\_gauss(a.copy(), identity\_row, n)  
 inverse\_matrix.append(inverse\_matrix\_col)  
  
 **return** transpose\_matrix(inverse\_matrix, n)  
  
  
**def** detailed\_solve\_by\_gauss(a, b, n):  
 *"""решение системы методом Гаусса (приведением к треугольному виду)"""* column = 0  
 unchanged\_a = a.copy()  
 unchanged\_b = b.copy()  
  
 **while** column < n:  
 print(**"Ищем максимальный по модулю элемент в {0}-м столбце:"**.format(column + 1))  
 current\_row = **None  
 for** r **in** range(column, n):  
 **if** current\_row **is None or** abs(a[r][column]) > abs(a[current\_row][column]):  
 current\_row = r  
 **if** current\_row **is None**:  
 print(**"решений нет"**)  
 **return None** format\_print(n, a, b, (current\_row, column))  
 **if** current\_row != column:  
 print(**"Переставляем строку с найденным элементом повыше:"**)  
 swap\_rows(a, b, current\_row, column)  
 format\_print(n, a, b, (column, column))  
 print(**f"Нормализуем строку с найденным элементом (делим на {**a[column][column]**}):"**)  
 **try**:  
 divide\_row(a, b, column, a[column][column])  
 **except** ZeroDivisionError:  
 **raise** ZeroDivisionError  
 format\_print(n, a, b, (column, column))  
 print(**"Обрабатываем нижележащие строки:"**)  
 **for** r **in** range(column + 1, n):  
 combine\_rows(a, b, r, column, -a[r][column])  
 format\_print(n, a, b, (column, column))  
 column += 1  
 print(**"Матрица приведена к треугольному виду, находим определитель"**)  
 det = calc\_det\_triangular\_matrix(a)  
 **if** det == 0:  
 print(**'Определитель равен нулю => матрица вырожденная'**)  
 **return None  
 else**:  
 print(**"Определитель равен {0} => матрица невырожденная, считаем решение"**.format(det))  
 x = [0 **for** \_ **in** b]  
 **for** i **in** range(n - 1, -1, -1):  
 x[i] = b[i] - sum(x \* a **for** x, a **in** zip(x[(i + 1):], a[i][(i + 1):])) *# zip создает итератор кортежей* print(**"\nПолучили ответ:"**)  
 print(**"\n"**.join(**"X{0} =\t{1:10.2f}"**.format(i + 1, x) **for** i, x **in** enumerate(x)))  
  
 list\_discrepancy = check\_discrepancy(unchanged\_a, unchanged\_b, x)  
 write\_output\_list(OUTPUT\_FILENAME, list\_discrepancy, **'Невязка'**)  
 inverse\_matrix = get\_inverse\_matrix(unchanged\_a, n)  
  
 print(**"\nОбратная матрица:"**)  
 format\_print(n, inverse\_matrix)  
 write\_output\_matrix(OUTPUT\_FILENAME, inverse\_matrix, **'Обратная матрица'**)  
 **return** x  
  
  
**def** calc\_det\_triangular\_matrix(matrix):  
 swap\_number = swap\_rows.count  
 det = (1 **if** swap\_number % 2 == 0 **else** -1)  
 **for** i **in** range(len(matrix)):  
 det \*= matrix[i][i]  
 **return** det  
  
  
**def** main():  
 swap\_number = 0  
 expanded\_matrix = list()  
 **with** open(OUTPUT\_FILENAME, mode=**"w"**, encoding=**'utf-8'**):  
 **pass  
 try**:  
 read\_input\_data(INPUT\_FILENAME, expanded\_matrix)  
 **except** Exception:  
 print(**"Ошибка при чтении файла"**)  
 **return** free\_factors = []  
 **for** \_row **in** expanded\_matrix:  
 free\_factors.append(\_row.pop())  
 factors\_at\_unknowns = expanded\_matrix  
 n = len(factors\_at\_unknowns)  
 print(**"\nИсходная система:"**)  
 format\_print(n, factors\_at\_unknowns, free\_factors, **None**)  
 print(**"\nРешаем:"**)  
  
 **try**:  
 detailed\_solve\_by\_gauss(factors\_at\_unknowns, free\_factors, n)  
 **except** ZeroDivisionError:  
 print(**"Определитель равен нулю!"**)  
  
 **try**:  
 write\_output\_matrix(OUTPUT\_FILENAME, expanded\_matrix, **'Расширенная матрица'**)  
 **except** Exception:  
 print(**"Ошибка при записи в файл"**)  
 **return  
  
  
if** \_\_name\_\_ == **'\_\_main\_\_'**:  
 main()

**Выводы**

Данная работа оказалась очень полезной в плане понимания программной реализации решения систем линейных уравнений методом Гаусса. Очень важно найти подходящий алгоритм решения, который бы удовлетворял временным и количественным требованиям решения задачи. Также очень важно вычислять и оценивать погрешность вычислений, так как даже небольшие погрешности могут привести к большим вычислительным ошибкам.

**Тест 1**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 2.00 7.00 4.00 9.00 ) = ( 6.00)

( -6.00 3.00 1.00 5.00 ) = ( 4.00)

( 3.00 8.00 5.00 6.00 ) = ( 7.00)

( 1.00 2.00 7.00 3.00 ) = ( 5.00)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 2.00 7.00 4.00 9.00 ) = ( 6.00)

( -6.00< 3.00 1.00 5.00 ) = ( 4.00)

( 3.00 8.00 5.00 6.00 ) = ( 7.00)

( 1.00 2.00 7.00 3.00 ) = ( 5.00)

Переставляем строку с найденным элементом повыше:

( -6.00< 3.00 1.00 5.00 ) = ( 4.00)

( 2.00 7.00 4.00 9.00 ) = ( 6.00)

( 3.00 8.00 5.00 6.00 ) = ( 7.00)

( 1.00 2.00 7.00 3.00 ) = ( 5.00)

Нормализуем строку с найденным элементом (делим на -6.0):

( 1.00< -0.50 -0.17 -0.83 ) = ( -0.67)

( 2.00 7.00 4.00 9.00 ) = ( 6.00)

( 3.00 8.00 5.00 6.00 ) = ( 7.00)

( 1.00 2.00 7.00 3.00 ) = ( 5.00)

Обрабатываем нижележащие строки:

( 1.00< -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 8.00 4.33 10.67 ) = ( 7.33)

( 0.00 9.50 5.50 8.50 ) = ( 9.00)

( 0.00 2.50 7.17 3.83 ) = ( 5.67)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 8.00 4.33 10.67 ) = ( 7.33)

( 0.00 9.50< 5.50 8.50 ) = ( 9.00)

( 0.00 2.50 7.17 3.83 ) = ( 5.67)

Переставляем строку с найденным элементом повыше:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 9.50< 5.50 8.50 ) = ( 9.00)

( 0.00 8.00 4.33 10.67 ) = ( 7.33)

( 0.00 2.50 7.17 3.83 ) = ( 5.67)

Нормализуем строку с найденным элементом (делим на 9.5):

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00< 0.58 0.89 ) = ( 0.95)

( 0.00 8.00 4.33 10.67 ) = ( 7.33)

( 0.00 2.50 7.17 3.83 ) = ( 5.67)

Обрабатываем нижележащие строки:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00< 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 -0.30 3.51 ) = ( -0.25)

( 0.00 0.00 5.72 1.60 ) = ( 3.30)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 -0.30 3.51 ) = ( -0.25)

( 0.00 0.00 5.72< 1.60 ) = ( 3.30)

Переставляем строку с найденным элементом повыше:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 5.72< 1.60 ) = ( 3.30)

( 0.00 0.00 -0.30 3.51 ) = ( -0.25)

Нормализуем строку с найденным элементом (делим на 5.719298245614035):

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 1.00< 0.28 ) = ( 0.58)

( 0.00 0.00 -0.30 3.51 ) = ( -0.25)

Обрабатываем нижележащие строки:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 1.00< 0.28 ) = ( 0.58)

( 0.00 0.00 0.00 3.59 ) = ( -0.07)

Ищем максимальный по модулю элемент в 4-м столбце:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 1.00 0.28 ) = ( 0.58)

( 0.00 0.00 0.00 3.59< ) = ( -0.07)

Нормализуем строку с найденным элементом (делим на 3.5920245398773):

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 1.00 0.28 ) = ( 0.58)

( 0.00 0.00 0.00 1.00< ) = ( -0.02)

Обрабатываем нижележащие строки:

( 1.00 -0.50 -0.17 -0.83 ) = ( -0.67)

( 0.00 1.00 0.58 0.89 ) = ( 0.95)

( 0.00 0.00 1.00 0.28 ) = ( 0.58)

( 0.00 0.00 0.00 1.00< ) = ( -0.02)

Матрица приведена к треугольному виду, находим определитель

Определитель равен -1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = -0.27

X2 = 0.63

X3 = 0.58

X4 = -0.02

Подсчёт невязки:

Невязка в 1-й строке = 0.0

Невязка в 2-й строке = -4.440892098500626e-16

Невязка в 3-й строке = 0.0

Невязка в 4-й строке = 8.881784197001252e-16

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 0.12 -0.15 -0.04 -0.02 )

( -0.20 0.07 0.31 -0.11 )

( -0.08 0.01 0.02 0.17 )

( 0.28 -0.02 -0.24 0.01 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

**Тест 2**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 1.00 3.00 8.00 ) = ( 25.00)

( 5.00 4.00 2.00 ) = ( 10.00)

( 10.00 5.00 1.00 ) = ( 15.00)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 1.00 3.00 8.00 ) = ( 25.00)

( 5.00 4.00 2.00 ) = ( 10.00)

( 10.00< 5.00 1.00 ) = ( 15.00)

Переставляем строку с найденным элементом повыше:

( 10.00< 5.00 1.00 ) = ( 15.00)

( 5.00 4.00 2.00 ) = ( 10.00)

( 1.00 3.00 8.00 ) = ( 25.00)

Нормализуем строку с найденным элементом (делим на 10.0):

( 1.00< 0.50 0.10 ) = ( 1.50)

( 5.00 4.00 2.00 ) = ( 10.00)

( 1.00 3.00 8.00 ) = ( 25.00)

Обрабатываем нижележащие строки:

( 1.00< 0.50 0.10 ) = ( 1.50)

( 0.00 1.50 1.50 ) = ( 2.50)

( 0.00 2.50 7.90 ) = ( 23.50)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 1.50 1.50 ) = ( 2.50)

( 0.00 2.50< 7.90 ) = ( 23.50)

Переставляем строку с найденным элементом повыше:

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 2.50< 7.90 ) = ( 23.50)

( 0.00 1.50 1.50 ) = ( 2.50)

Нормализуем строку с найденным элементом (делим на 2.5):

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 1.00< 3.16 ) = ( 9.40)

( 0.00 1.50 1.50 ) = ( 2.50)

Обрабатываем нижележащие строки:

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 1.00< 3.16 ) = ( 9.40)

( 0.00 0.00 -3.24 ) = ( -11.60)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 1.00 3.16 ) = ( 9.40)

( 0.00 0.00 -3.24< ) = ( -11.60)

Нормализуем строку с найденным элементом (делим на -3.24):

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 1.00 3.16 ) = ( 9.40)

( -0.00 -0.00 1.00< ) = ( 3.58)

Обрабатываем нижележащие строки:

( 1.00 0.50 0.10 ) = ( 1.50)

( 0.00 1.00 3.16 ) = ( 9.40)

( -0.00 -0.00 1.00< ) = ( 3.58)

Матрица приведена к треугольному виду, находим определитель

Определитель равен 1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = 2.10

X2 = -1.91

X3 = 3.58

Подсчёт невязки:

Невязка в 1-й строке = 0.0

Невязка в 2-й строке = 0.0

Невязка в 3-й строке = 1.7763568394002505e-15

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 0.07 -0.46 0.32 )

( -0.19 0.98 -0.47 )

( 0.19 -0.31 0.14 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

**Тест 3**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 1.00 8.00 4.00 5.00 6.00 ) = ( 10.00)

( 8.00 5.00 4.00 12.00 7.00 ) = ( 11.00)

( 3.00 15.00 4.00 5.00 4.00 ) = ( 20.00)

( 8.00 4.00 7.00 1.00 3.00 ) = ( 25.00)

( 4.00 8.00 7.00 2.00 4.00 ) = ( 5.00)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 1.00 8.00 4.00 5.00 6.00 ) = ( 10.00)

( 8.00< 5.00 4.00 12.00 7.00 ) = ( 11.00)

( 3.00 15.00 4.00 5.00 4.00 ) = ( 20.00)

( 8.00 4.00 7.00 1.00 3.00 ) = ( 25.00)

( 4.00 8.00 7.00 2.00 4.00 ) = ( 5.00)

Переставляем строку с найденным элементом повыше:

( 8.00< 5.00 4.00 12.00 7.00 ) = ( 11.00)

( 1.00 8.00 4.00 5.00 6.00 ) = ( 10.00)

( 3.00 15.00 4.00 5.00 4.00 ) = ( 20.00)

( 8.00 4.00 7.00 1.00 3.00 ) = ( 25.00)

( 4.00 8.00 7.00 2.00 4.00 ) = ( 5.00)

Нормализуем строку с найденным элементом (делим на 8.0):

( 1.00< 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 1.00 8.00 4.00 5.00 6.00 ) = ( 10.00)

( 3.00 15.00 4.00 5.00 4.00 ) = ( 20.00)

( 8.00 4.00 7.00 1.00 3.00 ) = ( 25.00)

( 4.00 8.00 7.00 2.00 4.00 ) = ( 5.00)

Обрабатываем нижележащие строки:

( 1.00< 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 7.38 3.50 3.50 5.12 ) = ( 8.62)

( 0.00 13.12 2.50 0.50 1.38 ) = ( 15.88)

( 0.00 -1.00 3.00 -11.00 -4.00 ) = ( 14.00)

( 0.00 5.50 5.00 -4.00 0.50 ) = ( -0.50)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 7.38 3.50 3.50 5.12 ) = ( 8.62)

( 0.00 13.12< 2.50 0.50 1.38 ) = ( 15.88)

( 0.00 -1.00 3.00 -11.00 -4.00 ) = ( 14.00)

( 0.00 5.50 5.00 -4.00 0.50 ) = ( -0.50)

Переставляем строку с найденным элементом повыше:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 13.12< 2.50 0.50 1.38 ) = ( 15.88)

( 0.00 7.38 3.50 3.50 5.12 ) = ( 8.62)

( 0.00 -1.00 3.00 -11.00 -4.00 ) = ( 14.00)

( 0.00 5.50 5.00 -4.00 0.50 ) = ( -0.50)

Нормализуем строку с найденным элементом (делим на 13.125):

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00< 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 7.38 3.50 3.50 5.12 ) = ( 8.62)

( 0.00 -1.00 3.00 -11.00 -4.00 ) = ( 14.00)

( 0.00 5.50 5.00 -4.00 0.50 ) = ( -0.50)

Обрабатываем нижележащие строки:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00< 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 2.10 3.22 4.35 ) = ( -0.30)

( 0.00 0.00 3.19 -10.96 -3.90 ) = ( 15.21)

( 0.00 0.00 3.95 -4.21 -0.08 ) = ( -7.15)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 2.10 3.22 4.35 ) = ( -0.30)

( 0.00 0.00 3.19 -10.96 -3.90 ) = ( 15.21)

( 0.00 0.00 3.95< -4.21 -0.08 ) = ( -7.15)

Переставляем строку с найденным элементом повыше:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 3.95< -4.21 -0.08 ) = ( -7.15)

( 0.00 0.00 3.19 -10.96 -3.90 ) = ( 15.21)

( 0.00 0.00 2.10 3.22 4.35 ) = ( -0.30)

Нормализуем строку с найденным элементом (делим на 3.9523809523809526):

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00< -1.07 -0.02 ) = ( -1.81)

( 0.00 0.00 3.19 -10.96 -3.90 ) = ( 15.21)

( 0.00 0.00 2.10 3.22 4.35 ) = ( -0.30)

Обрабатываем нижележащие строки:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00< -1.07 -0.02 ) = ( -1.81)

( 0.00 0.00 0.00 -7.56 -3.83 ) = ( 20.98)

( 0.00 0.00 0.00 5.45 4.39 ) = ( 3.50)

Ищем максимальный по модулю элемент в 4-м столбце:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00 -1.07 -0.02 ) = ( -1.81)

( 0.00 0.00 0.00 -7.56< -3.83 ) = ( 20.98)

( 0.00 0.00 0.00 5.45 4.39 ) = ( 3.50)

Нормализуем строку с найденным элементом (делим на -7.563855421686747):

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00 -1.07 -0.02 ) = ( -1.81)

( -0.00 -0.00 -0.00 1.00< 0.51 ) = ( -2.77)

( 0.00 0.00 0.00 5.45 4.39 ) = ( 3.50)

Обрабатываем нижележащие строки:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00 -1.07 -0.02 ) = ( -1.81)

( -0.00 -0.00 -0.00 1.00< 0.51 ) = ( -2.77)

( 0.00 0.00 0.00 0.00 1.63 ) = ( 18.62)

Ищем максимальный по модулю элемент в 5-м столбце:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00 -1.07 -0.02 ) = ( -1.81)

( -0.00 -0.00 -0.00 1.00 0.51 ) = ( -2.77)

( 0.00 0.00 0.00 0.00 1.63< ) = ( 18.62)

Нормализуем строку с найденным элементом (делим на 1.6301369863013697):

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00 -1.07 -0.02 ) = ( -1.81)

( -0.00 -0.00 -0.00 1.00 0.51 ) = ( -2.77)

( 0.00 0.00 0.00 0.00 1.00< ) = ( 11.42)

Обрабатываем нижележащие строки:

( 1.00 0.62 0.50 1.50 0.88 ) = ( 1.38)

( 0.00 1.00 0.19 0.04 0.10 ) = ( 1.21)

( 0.00 0.00 1.00 -1.07 -0.02 ) = ( -1.81)

( -0.00 -0.00 -0.00 1.00 0.51 ) = ( -2.77)

( 0.00 0.00 0.00 0.00 1.00< ) = ( 11.42)

Матрица приведена к треугольному виду, находим определитель

Определитель равен -1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = 8.09

X2 = 2.38

X3 = -10.71

X4 = -8.56

X5 = 11.42

Подсчёт невязки:

Невязка в 1-й строке = -7.105427357601002e-15

Невязка в 2-й строке = 0.0

Невязка в 3-й строке = 0.0

Невязка в 4-й строке = -2.1316282072803006e-14

Невязка в 5-й строке = -1.4210854715202004e-14

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 0.08 -0.03 0.09 0.31 -0.39 )

( 0.01 -0.04 0.11 0.04 -0.08 )

( -0.32 0.11 -0.15 -0.37 0.72 )

( -0.31 0.18 -0.04 -0.36 0.45 )

( 0.61 -0.17 -0.03 0.44 -0.68 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

Process finished with exit code 0

**Тест 4**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 4.00 2.00 -3.00 ) = ( 1.00)

( -8.00 -7.00 1.00 ) = ( 4.00)

( 4.00 2.00 -3.00 ) = ( 10.00)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 4.00 2.00 -3.00 ) = ( 1.00)

( -8.00< -7.00 1.00 ) = ( 4.00)

( 4.00 2.00 -3.00 ) = ( 10.00)

Переставляем строку с найденным элементом повыше:

( -8.00< -7.00 1.00 ) = ( 4.00)

( 4.00 2.00 -3.00 ) = ( 1.00)

( 4.00 2.00 -3.00 ) = ( 10.00)

Нормализуем строку с найденным элементом (делим на -8.0):

( 1.00< 0.88 -0.12 ) = ( -0.50)

( 4.00 2.00 -3.00 ) = ( 1.00)

( 4.00 2.00 -3.00 ) = ( 10.00)

Обрабатываем нижележащие строки:

( 1.00< 0.88 -0.12 ) = ( -0.50)

( 0.00 -1.50 -2.50 ) = ( 3.00)

( 0.00 -1.50 -2.50 ) = ( 12.00)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 0.88 -0.12 ) = ( -0.50)

( 0.00 -1.50< -2.50 ) = ( 3.00)

( 0.00 -1.50 -2.50 ) = ( 12.00)

Нормализуем строку с найденным элементом (делим на -1.5):

( 1.00 0.88 -0.12 ) = ( -0.50)

( -0.00 1.00< 1.67 ) = ( -2.00)

( 0.00 -1.50 -2.50 ) = ( 12.00)

Обрабатываем нижележащие строки:

( 1.00 0.88 -0.12 ) = ( -0.50)

( -0.00 1.00< 1.67 ) = ( -2.00)

( 0.00 0.00 0.00 ) = ( 9.00)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 0.88 -0.12 ) = ( -0.50)

( -0.00 1.00 1.67 ) = ( -2.00)

( 0.00 0.00 0.00< ) = ( 9.00)

Нормализуем строку с найденным элементом (делим на 0.0):

Определитель равен нулю!

Запись в файл проведена успешно (Расширенная матрица)

**Тест 5**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 1.00 5.00 4.00 ) = ( 10.00)

( 2.00 8.00 4.00 ) = ( 20.00)

( 4.00 9.00 1.00 ) = ( 15.00)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 1.00 5.00 4.00 ) = ( 10.00)

( 2.00 8.00 4.00 ) = ( 20.00)

( 4.00< 9.00 1.00 ) = ( 15.00)

Переставляем строку с найденным элементом повыше:

( 4.00< 9.00 1.00 ) = ( 15.00)

( 2.00 8.00 4.00 ) = ( 20.00)

( 1.00 5.00 4.00 ) = ( 10.00)

Нормализуем строку с найденным элементом (делим на 4.0):

( 1.00< 2.25 0.25 ) = ( 3.75)

( 2.00 8.00 4.00 ) = ( 20.00)

( 1.00 5.00 4.00 ) = ( 10.00)

Обрабатываем нижележащие строки:

( 1.00< 2.25 0.25 ) = ( 3.75)

( 0.00 3.50 3.50 ) = ( 12.50)

( 0.00 2.75 3.75 ) = ( 6.25)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 2.25 0.25 ) = ( 3.75)

( 0.00 3.50< 3.50 ) = ( 12.50)

( 0.00 2.75 3.75 ) = ( 6.25)

Нормализуем строку с найденным элементом (делим на 3.5):

( 1.00 2.25 0.25 ) = ( 3.75)

( 0.00 1.00< 1.00 ) = ( 3.57)

( 0.00 2.75 3.75 ) = ( 6.25)

Обрабатываем нижележащие строки:

( 1.00 2.25 0.25 ) = ( 3.75)

( 0.00 1.00< 1.00 ) = ( 3.57)

( 0.00 0.00 1.00 ) = ( -3.57)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 2.25 0.25 ) = ( 3.75)

( 0.00 1.00 1.00 ) = ( 3.57)

( 0.00 0.00 1.00< ) = ( -3.57)

Нормализуем строку с найденным элементом (делим на 1.0):

( 1.00 2.25 0.25 ) = ( 3.75)

( 0.00 1.00 1.00 ) = ( 3.57)

( 0.00 0.00 1.00< ) = ( -3.57)

Обрабатываем нижележащие строки:

( 1.00 2.25 0.25 ) = ( 3.75)

( 0.00 1.00 1.00 ) = ( 3.57)

( 0.00 0.00 1.00< ) = ( -3.57)

Матрица приведена к треугольному виду, находим определитель

Определитель равен -1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = -11.43

X2 = 7.14

X3 = -3.57

Подсчёт невязки:

Невязка в 1-й строке = -3.552713678800501e-15

Невязка в 2-й строке = 0.0

Невязка в 3-й строке = -1.7763568394002505e-15

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 2.00 -2.21 0.86 )

( -1.00 1.07 -0.29 )

( 1.00 -0.79 0.14 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

Process finished with exit code 0

**Тест 6**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 7.00 10.00 8.00 7.00 ) = ( 32.00)

( 6.00 8.00 10.00 9.00 ) = ( 33.00)

( 5.00 7.00 9.00 10.00 ) = ( 31.00)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 7.00< 10.00 8.00 7.00 ) = ( 32.00)

( 6.00 8.00 10.00 9.00 ) = ( 33.00)

( 5.00 7.00 9.00 10.00 ) = ( 31.00)

Переставляем строку с найденным элементом повыше:

( 7.00< 10.00 8.00 7.00 ) = ( 32.00)

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 6.00 8.00 10.00 9.00 ) = ( 33.00)

( 5.00 7.00 9.00 10.00 ) = ( 31.00)

Нормализуем строку с найденным элементом (делим на 7.0):

( 1.00< 1.43 1.14 1.00 ) = ( 4.57)

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 6.00 8.00 10.00 9.00 ) = ( 33.00)

( 5.00 7.00 9.00 10.00 ) = ( 31.00)

Обрабатываем нижележащие строки:

( 1.00< 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.14)

( 0.00 -0.57 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 3.29 5.00 ) = ( 8.14)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.14)

( 0.00 -0.57< 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 3.29 5.00 ) = ( 8.14)

Переставляем строку с найденным элементом повыше:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.57< 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.14)

( 0.00 -0.14 3.29 5.00 ) = ( 8.14)

Нормализуем строку с найденным элементом (делим на -0.5714285714285712):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00< -5.50 -5.25 ) = ( -9.75)

( 0.00 -0.14 0.29 0.00 ) = ( 0.14)

( 0.00 -0.14 3.29 5.00 ) = ( 8.14)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00< -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.25)

( 0.00 0.00 2.50 4.25 ) = ( 6.75)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.25)

( 0.00 0.00 2.50< 4.25 ) = ( 6.75)

Переставляем строку с найденным элементом повыше:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 2.50< 4.25 ) = ( 6.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.25)

Нормализуем строку с найденным элементом (делим на 2.4999999999999982):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00< 1.70 ) = ( 2.70)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.25)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00< 1.70 ) = ( 2.70)

( 0.00 0.00 0.00 0.10 ) = ( 0.10)

Ищем максимальный по модулю элемент в 4-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.70)

( 0.00 0.00 0.00 0.10< ) = ( 0.10)

Нормализуем строку с найденным элементом (делим на 0.10000000000000098):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.70)

( 0.00 0.00 0.00 1.00< ) = ( 1.00)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.70)

( 0.00 0.00 0.00 1.00< ) = ( 1.00)

Матрица приведена к треугольному виду, находим определитель

Определитель равен -1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = 1.00

X2 = 1.00

X3 = 1.00

X4 = 1.00

Подсчёт невязки:

Невязка в 1-й строке = 0.0

Невязка в 2-й строке = -3.552713678800501e-15

Невязка в 3-й строке = 0.0

Невязка в 4-й строке = 0.0

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 68.00 -41.00 -17.00 10.00 )

( -41.00 25.00 10.00 -6.00 )

( -17.00 10.00 5.00 -3.00 )

( 10.00 -6.00 -3.00 2.00 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

Process finished with exit code 0

**Тест 7**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 7.00 10.00 8.00 7.00 ) = ( 31.99)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 7.00< 10.00 8.00 7.00 ) = ( 31.99)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Переставляем строку с найденным элементом повыше:

( 7.00< 10.00 8.00 7.00 ) = ( 31.99)

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Нормализуем строку с найденным элементом (делим на 7.0):

( 1.00< 1.43 1.14 1.00 ) = ( 4.57)

( 5.00 7.00 6.00 5.00 ) = ( 23.00)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Обрабатываем нижележащие строки:

( 1.00< 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.15)

( 0.00 -0.57 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.15)

( 0.00 -0.57< 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Переставляем строку с найденным элементом повыше:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.57< 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.15)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Нормализуем строку с найденным элементом (делим на -0.5714285714285712):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00< -5.50 -5.25 ) = ( -9.75)

( 0.00 -0.14 0.29 0.00 ) = ( 0.15)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00< -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.24)

( 0.00 0.00 2.50 4.25 ) = ( 6.77)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.24)

( 0.00 0.00 2.50< 4.25 ) = ( 6.77)

Переставляем строку с найденным элементом повыше:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 2.50< 4.25 ) = ( 6.77)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.24)

Нормализуем строку с найденным элементом (делим на 2.4999999999999982):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00< 1.70 ) = ( 2.71)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.24)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00< 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 0.10 ) = ( 0.11)

Ищем максимальный по модулю элемент в 4-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 0.10< ) = ( 0.11)

Нормализуем строку с найденным элементом (делим на 0.10000000000000098):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 1.00< ) = ( 1.11)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 1.00< ) = ( 1.11)

Матрица приведена к треугольному виду, находим определитель

Определитель равен -1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = 1.68

X2 = 0.59

X3 = 0.82

X4 = 1.11

Подсчёт невязки:

Невязка в 1-й строке = -3.552713678800501e-15

Невязка в 2-й строке = -3.552713678800501e-15

Невязка в 3-й строке = 0.0

Невязка в 4-й строке = -3.552713678800501e-15

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 68.00 -41.00 -17.00 10.00 )

( -41.00 25.00 10.00 -6.00 )

( -17.00 10.00 5.00 -3.00 )

( 10.00 -6.00 -3.00 2.00 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

Process finished with exit code 0

**Тест 8**

C:\Python38-32\python.exe C:/Users/Victor/MyFolder/AltSTU\_repo/6\_sem/Comp-algorithms\_\_Pyshogray-Proskurin/labs/lab1/main.py

Чтение из файла проведено успешно

Исходная система:

( 5.00 7.00 6.00 5.00 ) = ( 23.10)

( 7.00 10.00 8.00 7.00 ) = ( 31.99)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Решаем:

Ищем максимальный по модулю элемент в 1-м столбце:

( 5.00 7.00 6.00 5.00 ) = ( 23.10)

( 7.00< 10.00 8.00 7.00 ) = ( 31.99)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Переставляем строку с найденным элементом повыше:

( 7.00< 10.00 8.00 7.00 ) = ( 31.99)

( 5.00 7.00 6.00 5.00 ) = ( 23.10)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Нормализуем строку с найденным элементом (делим на 7.0):

( 1.00< 1.43 1.14 1.00 ) = ( 4.57)

( 5.00 7.00 6.00 5.00 ) = ( 23.10)

( 6.00 8.00 10.00 9.00 ) = ( 32.99)

( 5.00 7.00 9.00 10.00 ) = ( 31.01)

Обрабатываем нижележащие строки:

( 1.00< 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.25)

( 0.00 -0.57 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Ищем максимальный по модулю элемент в 2-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.25)

( 0.00 -0.57< 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Переставляем строку с найденным элементом повыше:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( 0.00 -0.57< 3.14 3.00 ) = ( 5.57)

( 0.00 -0.14 0.29 0.00 ) = ( 0.25)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Нормализуем строку с найденным элементом (делим на -0.5714285714285712):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00< -5.50 -5.25 ) = ( -9.75)

( 0.00 -0.14 0.29 0.00 ) = ( 0.25)

( 0.00 -0.14 3.29 5.00 ) = ( 8.16)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00< -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.14)

( 0.00 0.00 2.50 4.25 ) = ( 6.77)

Ищем максимальный по модулю элемент в 3-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.14)

( 0.00 0.00 2.50< 4.25 ) = ( 6.77)

Переставляем строку с найденным элементом повыше:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 2.50< 4.25 ) = ( 6.77)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.14)

Нормализуем строку с найденным элементом (делим на 2.4999999999999982):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00< 1.70 ) = ( 2.71)

( 0.00 0.00 -0.50 -0.75 ) = ( -1.14)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00< 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 0.10 ) = ( 0.21)

Ищем максимальный по модулю элемент в 4-м столбце:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 0.10< ) = ( 0.21)

Нормализуем строку с найденным элементом (делим на 0.10000000000000098):

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 1.00< ) = ( 2.11)

Обрабатываем нижележащие строки:

( 1.00 1.43 1.14 1.00 ) = ( 4.57)

( -0.00 1.00 -5.50 -5.25 ) = ( -9.75)

( 0.00 0.00 1.00 1.70 ) = ( 2.71)

( 0.00 0.00 0.00 1.00< ) = ( 2.11)

Матрица приведена к треугольному виду, находим определитель

Определитель равен -1.0 => матрица невырожденная, считаем решение

Получили ответ:

X1 = 8.48

X2 = -3.51

X3 = -0.88

X4 = 2.11

Подсчёт невязки:

Невязка в 1-й строке = -1.0658141036401503e-14

Невязка в 2-й строке = -1.0658141036401503e-14

Невязка в 3-й строке = -7.105427357601002e-15

Невязка в 4-й строке = -1.0658141036401503e-14

Запись в файл проведена успешно (Невязка)

Обратная матрица:

( 68.00 -41.00 -17.00 10.00 )

( -41.00 25.00 10.00 -6.00 )

( -17.00 10.00 5.00 -3.00 )

( 10.00 -6.00 -3.00 2.00 )

Запись в файл проведена успешно (Обратная матрица)

Запись в файл проведена успешно (Расширенная матрица)

Process finished with exit code 0