Федеральное государственное автономное образовательное учреждение высшего образования

***Университет ИТМО***

Факультет программной инженерии и компьютерной техники

Дисциплина: **Вычислительная математика**

Лабораторная работа №1

*“* **Решение системы линейных алгебраических уравнений СЛАУ***”*

**Вариант: 9 (Теорема Гаусса)**

Выполнил: Кузнецов Максим Александрович

Группа: Р3111

Преподаватель: Малышева Татьяна Алексеевна

Санкт-Петербург 2022 г

# Цель работы:

Изучить численные методы решения систем линейных алгебраических уравнений и реализовать один из них (по варианту *Метод Гаусса с выбором главного элемента по столбцам*) средствами программирования

Изучить различные методы и способы решения СЛАУ с использованием языка программирования, по выданному варианту (Метод Гаусса) реализовать один из них на любом языке программирования (Python в данном случае).

**Условия:**

1. № варианта определяется как номер в списке группы согласно ИСУ.
2. В программе численный метод должен быть реализован в виде отдельной подпрограммы или класса, в который входные/выходные данные передаются в качестве параметров.
3. Размерность матрицы n <= 20 (задается из файла или с клавиатуры - по выбору конечного пользователя).
4. Должна быть реализована возможность ввода коэффициентов матрицы, как с клавиатуры, так и из файла (по выбору конечного пользователя).

**Для прямых методов должно быть реализовано:**

* Вычисление определителя
* Вывод треугольной матрицы (включая преобразованный столбец В)
* Вывод вектора неизвестных:
* Вывод вектора невязок:

# Описание метода:

Основан на приведении матрицы системы к треугольному виду так, чтобы ниже ее главной диагонали находились только нулевые элементы.

Прямой ход метода Гаусса состоит в последовательном исключении неизвестных из уравнений системы. Сначала с помощью первого уравнения исключается 𝑥1 из всех последующих уравнений системы. Затем с помощью второго уравнения исключается 𝑥2 из третьего и всех последующих уравнений и т.д.

Этот процесс продолжается до тех пор, пока в левой части последнего (n-го) уравнения не останется лишь один член с неизвестным 𝑥𝑛 , т. е. матрица системы будет приведена к треугольному виду.

Обратный ход метода Гаусса состоит в последовательном вычислении искомых неизвестных: решая последнее уравнение, находим единственное в этом уравнении неизвестное 𝑥𝑛. Далее, используя это значение, из предыдущего уравнения вычисляем 𝑥𝑛−1 и т. д. Последним найдем 𝑥1 из первого уравнения.

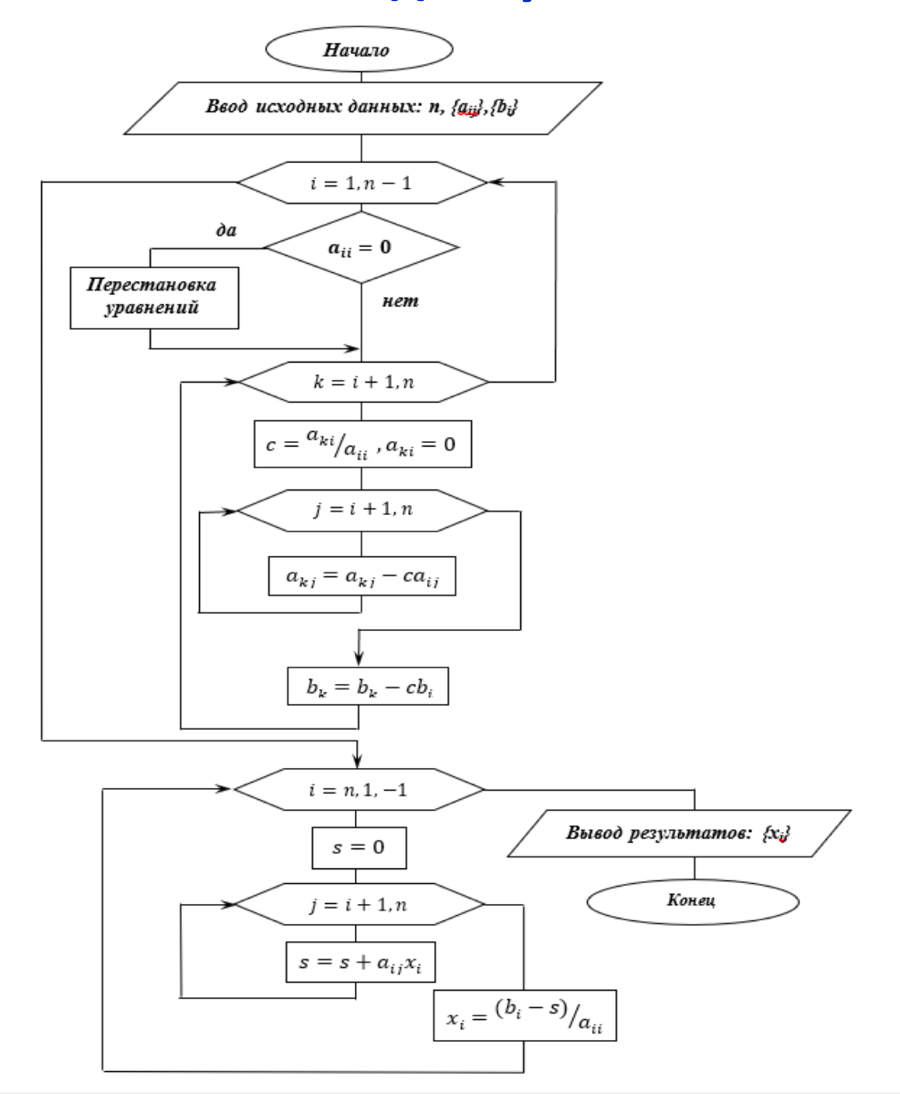
Метод имеет много различных вычислительных схем, но в каждой из них основным требованием является det A ≠ 0 .

Если в процессе исключения неизвестных, коэффициенты:

a11, a22, a33 . . . = 0

тогда необходимо соответственным образом переставить уравнения системы.

Перестановка уравнений должна быть предусмотрена в вычислительном алгоритме при его реализации на компьютере.



*Примечание:*

В моей программе это просто поиск максимального элемента по столбцу, фактически, можно было бы и искать ближайший, неравный нулю, но особо роли это не сыграло.

# Листинг программы:

import numpy as np  
import sys  
import copy  
  
INPUT = "input.txt"  
A = []  
residuals = []  
  
  
def isint(s):  
 try:  
 int(s)  
 return True  
 except ValueError:  
 return False  
  
  
def print\_data(mat):  
 for i in range(number):  
 print('X', end='')  
 print(i + 1, end='')  
 print('=', end=' ')  
 s = '{0:.100f}'.format(mat[i]).rstrip('0').rstrip('.')  
 conter = 0  
 if (isint(s)):  
 print(mat[i], end='\t')  
 else:  
 first = True  
 print(s.split('.')[0], end='')  
 print('.', end='')  
 for c in s.split('.')[1]:  
 if c != '0':  
 first = False  
 conter += 1  
 print(c, end='')  
 if conter == 3:  
 break  
 else:  
 if first:  
 print('0', end='')  
 else:  
 conter += 1  
 print(c, end='')  
 if conter == 3:  
 break  
 print(' ', end='')  
 print()  
  
  
def print\_matrix(mat):  
 for row in mat:  
 for value in row:  
 print('{}'.format(round(value, 3)), end='\t')  
 print()  
  
  
def check\_matrix(det):  
 if det == 0:  
 print("\nСистема не имеет решение или имеет бесконечное множество решений!")  
 sys.exit()  
  
  
def read\_from\_file():  
 with open(INPUT, 'r', encoding='utf-8') as input:  
 try:  
 temp = []  
 global number  
 number = int(input.readline())  
 lines = input.readlines()  
 for r in lines:  
 row = []  
 coefs = r.strip().split()  
 counter = -1  
 for coef in coefs:  
 counter += 1  
 row.append(float(coef))  
 if counter == (number - 1):  
 A.append(copy.copy(row))  
 if len(row) != (number + 1):  
 raise ValueError  
 temp.append(row)  
 if len(temp) != number:  
 raise ValueError  
 except ValueError:  
 print("\nНеверно составлен файл.")  
 return None  
 return temp  
  
  
def read\_from\_console():  
 while True:  
 n = int(input("\nРазмерность матрицы: "))  
 if n <= 0:  
 print("\nПорядок > 0!")  
 elif n > 20:  
 print("\nПорядок <= 20!")  
 else:  
 break  
 temp = []  
 print("\nКоэффициенты матрицы (через пробел): ")  
 try:  
 for i in range(n):  
 row = []

counter = -1  
 coefs = input().strip().split()  
 for coef in coefs:

counter += 1  
 row.append(float(coef))

if counter == (number - 1):  
 A.append(copy.copy(row))  
 if len(row) != (n + 1):  
 raise ValueError  
 temp.append(row)  
 except ValueError:  
 print("\nНеверно введены данные.")  
 return None  
 return temp  
  
  
def swap\_rows(m, col):  
 max\_element = m[col][col]  
 max\_row = col  
 for i in range(col + 1, len(m)):  
 if abs(m[i][col]) > abs(max\_element):  
 max\_element = m[i][col]  
 max\_row = i  
 if max\_row != col:  
 m[col], m[max\_row] = m[max\_row], m[col]  
  
  
def gauss\_method():  
 for i in range(number):  
 if matrix[i][i] == 0.0:  
 swap\_rows(matrix, i)  
 for j in range(i + 1, number):  
 ratio = matrix[j][i] / matrix[i][i]  
 for k in range(number + 1):  
 matrix[j][k] = matrix[j][k] - ratio \* matrix[i][k]  
  
  
def back\_substitution():  
 roots[number - 1] = matrix[number - 1][number] / matrix[number - 1][number - 1]  
 for i in range(number - 2, -1, -1):  
 roots[i] = matrix[i][number]  
 for j in range(i + 1, number):  
 roots[i] = roots[i] - matrix[i][j] \* roots[j]  
 roots[i] = roots[i] / matrix[i][i]  
  
  
def count\_residuals():  
 global residuals  
 residuals = [0] \* number  
 for i in range(number):  
 calculated\_part = 0  
  
 for j in range(number):  
 calculated\_part += matrix[i][j] \* roots[j]  
  
 residuals[i] = calculated\_part - matrix[i][number]  
  
  
while True:  
 method = input("\nУкажите способ (файл - 0, консоль - 1):")  
 global matrix  
 if method == "0":  
 matrix = read\_from\_file()  
 break  
 elif method == "1":  
 matrix = read\_from\_console()  
 break  
 else:  
 continue  
  
original\_matrix = copy.copy(matrix)  
print("\nИсходная матрица:")  
print\_matrix(matrix)  
roots = np.zeros(number)  
print("\nОпределитель:")  
det = np.linalg.det(A)  
check\_matrix(det)  
print(det)  
print("\nПриведенная матрица:")  
gauss\_method()  
print\_matrix(matrix)  
print("\nКорни:")  
back\_substitution()  
print\_data(roots)  
print("\nВектор невязок:")  
count\_residuals()  
print(residuals)

# Примечание о коде:

1. Вывод корней немного отличается от обычного print. По требованию практика была реализована возможность вывода до трех значащих цифр, а не простое округление до трех цифр. Это позволяет выводить числа разной длины, но совершенно точно, без ошибок.
2. Для нахождения детерминанта разрешено было использовать библиотеку Python “numpy”, собственно, с помощью нее в два действия и находиться определитель, который и играет дальнейшую роль в программе.

# Пример работы:

Изображение выглядит как текст

Автоматически созданное описание

# Вывод:

В результате выполнения лабораторной работы я:

* Познакомился с различными способами решения СЛАУ, с использованием языка программирования
* На практике воспользовался изученными ранее математическими «премудростями»
* Поработал с Python