БЕЛОРУССКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ   
Факультет прикладной математики и информатики

**Лабораторная работа №1**   
по теме  
«Метод Гаусса с выбором главного элемента по столбцу для численного решения систем линейных алгебраических уравнений»

Выполнила   
Молочко Екатерина  
2 курс 7 группа

Преподаватель  
Горбачева Юлия Николаевна

Минск   
2021

# Постановка задачи

1. Найти решение системы линейных алгебраических уравнений с помощью метода Гаусса с выбором главного элемента по столбцу
2. Вычислить обратную матрицу
3. Вычислить погрешность решения системы линейных алгебраических уравнений
4. Оценить точность решения системы линейных алгебраических уравнений

# Теоретические сведения

Пусть

система неоднородных линейных уравнений, в таком случае можно представить систему в виде произведения матриц , где

A – матрица коэффициентов  
x – столбец неизвестных   
f – столбец свободных членов

или расширенной матрицы системы

Элементарные преобразования линейной системы:

1. умножение какого-либо уравнения на ненулевой скаляр
2. прибавление к какому-либо уравнению системы другого уравнения, умноженного на ненулевой скаляр

## Алгоритм нахождения решения линейной неоднородной системы методом Гаусса с выбором главного элемента по столбцу

1. Рассмотрим 1-й столбец матрицы A, и найдем максимальный по модулю элемент, пусть этот элемент , (если такой элемент равен 0, то система с n неизвестными не имеет решения)
2. Поменяем местами 1-ю и i-ую строку
3. Поделим новую первую строку на элемент
4. Умножим первую строку на элемент и отнимем от 2-ой строкиПроделываем данное действие со всеми строками
5. Повторяем алгоритм с самого начала, рассматривая подматрицы k-го порядка
6. В результате получим матрицу

## Обратный ход метода Гаусса

Заметим, что  
 =>   
 =>   
продолжая разворачивать систему, найдем вектор решения

Для вычисления обратной матрицы, мы используем прямой и обратный ход метода Гаусса – Жордана, аналогичный прямому и обратному ходу метода Гаусса, различием является зануление столбцов над главной диагональю.

# Листинг программы

**import** numpy

**import** numpy **as** np

**def** max\_in\_column**(**matrix**,** degree**,** terms**):**

step **=** np**.abs(**degree **-** **len(**matrix**))**

matrix\_ **=** np**.abs(**matrix**)**

max\_i **=** np**.**argmax**(**matrix\_**[**step**:,** step**])** **+** step

**if** matrix**[**max\_i**,** step**]** **==** 0**:**

**print(**"Modulo maximum element in column is zero. Gaussian method can't be applied."**)**

**return**

**if** max\_i **==** step**:**

**return** matrix**[**step**,** step**]**

**if** max\_i **!=** step**:**

matrix**[**step**,** **:],** matrix**[**max\_i**,** **:]** **=** np**.**copy**(**matrix**[**max\_i**,** **:]),** np**.**copy**(**matrix**[**step**,** **:])**

terms**[**step**],** terms**[**max\_i**]** **=** np**.**copy**(**terms**[**max\_i**]),** np**.**copy**(**terms**[**step**])**

maximal **=** matrix**[**step**,** step**]**

**return** maximal

# деление первой строки подматрицы на первый элемент

**def** division**(**matrix**,** degree**,** terms**):**

step **=** np**.abs(**np**.**copy**(**degree**)** **-** **len(**matrix**))**

divider **=** max\_in\_column**(**matrix**,** degree**,** terms**)**

**float(**divider**)**

**for** i **in** **range(**step**,** **len(**matrix**[**degree **-** 1**])):**

matrix**[**step**,** i**]** **/=** divider

terms**[**step**]** **/=** divider

# умножение строки на 1-ый элемент каждой строки подматрицы

**def** mult\_sub**(**matrix**,** degree**,** terms**):**

# и вычитание из каждой строки первой домноженной

step **=** np**.abs(**np**.**copy**(**degree**)** **-** **len(**matrix**))**

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

multiplier **=** matrix**[**i**,** step**]**

subtracted **=** np**.**multiply**(**matrix**[**step**,** step**:],** multiplier**)**

subtracted\_t **=** terms**[**step**]** **\*** multiplier

terms**[**i**]** **-=** subtracted\_t

matrix**[**i**,** step**:]** **=** matrix**[**i**,** step**:]** **-** subtracted

# прямой ход метода Гаусса

**def** gaussian\_elimination**(**matrix**,** degree**,** terms**):**

**for** k **in** **range(**degree**,** 1**,** **-**1**):**

max\_in\_column**(**matrix**,** k**,** terms**)**

division**(**matrix**,** k**,** terms**)**

mult\_sub**(**matrix**,** k**,** terms**)**

division**(**matrix**,** 1**,** terms**)**

# обратный ход

**def** forward\_elimination**(**matrix**,** degree**,** terms**):**

X\_ **=** np**.**array**([**0.0**]** **\*** degree**)**

X\_**[**degree **-** 1**]** **=** terms**[**degree **-** 1**]**

**for** i **in** **range(**degree **-** 2**,** **-**1**,** **-**1**):**

**sum** **=** 0

**for** j **in** **range(**degree **-** 1**,** 0**,** **-**1**):**

**sum** **+=** matrix**[**i**,** j**]** **\*** X\_**[**j**]**

X\_**[**i**]** **=** terms**[**i**]** **-** np**.**copy**(sum)**

**return** X\_

**def** inverted\_matrix**(**matrix**,** degree**):**

E **=** np**.**diag**([**1 **for** i **in** **range(**degree**)]).**astype**(float)**

wide **=** np**.**hstack**((**matrix**,** E**)).**astype**(float)**

**for** i **in** **range(**degree**,** 0**,** **-**1**):**

max\_in\_column\_i**(**wide**,** i**)**

**for** j **in** **range(**0**,** degree**):**

**if** wide**[**j**,** j**]** **!=** 1.**:**

wide**[**j**,** **:]** **\*=** **float(**1.0 **/** wide**[**j**,** j**])**

**for** row **in** **range(**j **+** 1**,** degree**):**

wide**[**row**,** **:]** **-=** wide**[**j**,** **:]** **\*** wide**[**row**,** j**]**

**for** i **in** **range(**degree **-** 1**,** 0**,** **-**1**):**

# и тут мб range не тот

**for** row **in** **range(**i **-** 1**,** **-**1**,** **-**1**):**

**if** **(**wide**[**row**,** i**]):**

wide**[**row**,** **:]** **-=** wide**[**i**,** **:]** **\*** wide**[**row**,** i**]**

**return** np**.**hsplit**(**wide**,** degree **//** 2**)[**1**]**

**def** max\_in\_column\_i**(**wide**,** degree**):**

step **=** np**.abs(**degree **-** wide**.**shape**[**0**])**

matrix\_ **=** np**.abs(**wide**)**

**print(**"step \n"**,** step**)**

**print(**matrix\_**[**step**:,** step**])**

max\_i **=** np**.**argmax**(**matrix\_**[**step**:,** step**])** **+** step

**if** wide**[**max\_i**,** step**]** **==** 0**:**

**print(**"Modulo maximum element in column is zero. Gaussian method can't be applied."**)**

**return**

**if** max\_i **==** step**:**

**return** wide**[**step**,** step**]**

**if** max\_i **!=** step**:**

wide**[**step**,** **:],** wide**[**max\_i**,** **:]** **=** np**.**copy**(**wide**[**max\_i**,** **:]),** np**.**copy**(**wide**[**step**,** **:])**

maximal **=** wide**[**step**,** step**]**

**return** maximal

k **=** **int(input(**'Degree of a matrix: '**))**

# создание квадратной матрицы размера k согласно условию

A **=** np**.**random**.**randint**(-**100**,** 100**,** **(**k**,** k**)).**astype**(float)**

**print(**'\nSystem matrix is: \n'**,** A**)**

# создание вектора неизвестных

X **=** np**.**array**([**0**]** **\*** k**)**

**for** i **in** **range(len(**X**)):**

X**[**i**]** **+=** i **+** 1

**print(**'\nVector of real unknowns: \n'**,** X**)**

# перемножение вектора неизвестных и матрицы для получения свободных членов

B **=** numpy**.**matmul**(**A**,** X**)**

**print(**'\nConstant terms column: \n'**,** B**)**

gaussian\_elimination**(**A**,** k**,** B**)**

X\_ **=** forward\_elimination**(**A**,** k**,** B**)**

**print(**"\nVector of computed unknowns "**,** X\_**)**

format\_string **=** "{:.16f}"

**for** i **in** X\_**:**

**print(**format\_string**.format(**i**))**

# вычисление максимум-норм векторов

v\_norm\_sub **=** np**.**array**(**np**.abs(**np**.**subtract**(**X**,** X\_**)))**

v\_norm\_unk **=** np**.abs(**X**)**

# вычиление погрешности

uncertainty **=** v\_norm\_sub**[**np**.**argmax**(**v\_norm\_sub**)]** **/** v\_norm\_unk**[**np**.**argmax**(**v\_norm\_unk**)]**

**print(**"\nUncertainty is"**,** uncertainty**)**

inver **=** np**.**random**.**randint**(-**100**,** 100**,** **(**k**,** k**))**

**print(**'\n'**,** inver**)**

inversed **=** inverted\_matrix**(**inver**,** k**)**

**print(**inversed**)**

**print(**np**.**matmul**(**inver**,** inversed**))**

# Результаты

|  |  |  |  |
| --- | --- | --- | --- |
| порядок |  | порядок |  |
| 8 | 4.884981308350689e-15 | 68 | 7.523393672754002e-15 |
| 18 | 1.973729821555834e-15 | 78 | 4.4454468596272933e-13 |
| 28 | 6.851662094829538e-15 | 88 | 1.5244371421762149e-13 |
| 38 | 4.814861959426995e-15 | 98 | 9.425566902940104e-15 |
| 48 | 5.0774199659523824e-14 | 108 | 1.2684503653198825e-13 |
| 58 | 2.4623981015134508e-14 |  |  |

# Вывод:

Заметно, что при повышении размерности системы увеличивается относительная погрешность. Относительная погрешность является достаточно малой, для машинного вычисления решения систем линейных уравнений методом Гаусса.