Министерство образования Республики Беларусь

Учреждение образования «Белорусский государственный университет

информатики и радиоэлектроники»

Факультет компьютерных систем и сетей

Кафедра информатики

Дисциплина «Методы численного анализа»

**ОТЧЕТ**

к лабораторной работе №1

на тему:

**«Решение систем линейных алгебраических уравнний (СЛАУ)**

**методом Гаусса и с помощью его модификаций»**

БГУИР 1-40-04-01

|  |
| --- |
| Выполнил студент группы 253504  Дмитрук Богдан Ярославович |
|  |
| (дата, подпись студента) |
| Проверил  Анисимов Владимир Яковлевич |
|  |
| (дата, подпись преподавателя) |

Минск 2023

# **СОДЕРЖАНИЕ**

[**1. Цели выполнения задания**](#_vooml75z5ldu) **2**

[**2. Краткие теоретические сведения**](#_kttzpxlzwhrq) **5**

3. [Задание 9](file:///C:\Users\user\Downloads\Telegram%20Desktop\lab1_report.docx#_Toc145188322)

4.[Программная реализация 10](file:///C:\Users\user\Downloads\Telegram%20Desktop\lab1_report.docx#_Toc145188323)

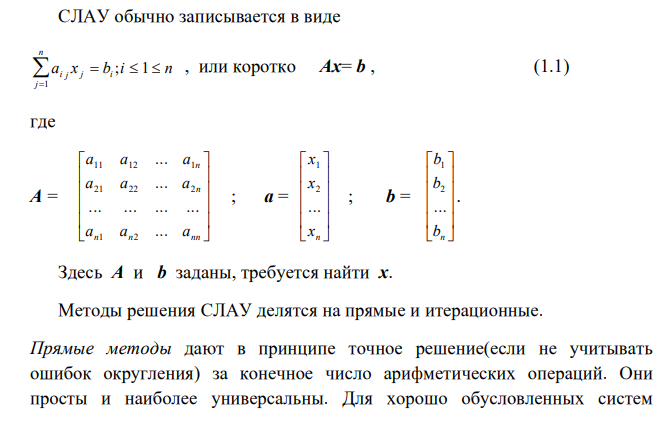
5. [Полученные результаты 19](file:///C:\Users\user\Downloads\Telegram%20Desktop\lab1_report.docx#_Toc145188324)

6. [Оценка 23](file:///C:\Users\user\Downloads\Telegram%20Desktop\lab1_report.docx#_Toc145188325)

7. [Выводы 24](file:///C:\Users\user\Downloads\Telegram%20Desktop\lab1_report.docx#_Toc145188326)

# **ЦЕЛИ ВЫПОЛНЕНИЯ ЗАДАНИЯ**

* Изучить метод Гаусса и его модификации, составить алгоритм метода и программу его реализации, получить численное решение заданной СЛАУ;
* составить алгоритм решения СЛАУ указанными методами, применимый для организации вычислений на ЭВМ;
* составить программу решения СЛАУ по разработанному алгоритму;
* выполнить тестовые примеры и проверить правильность работы программы.

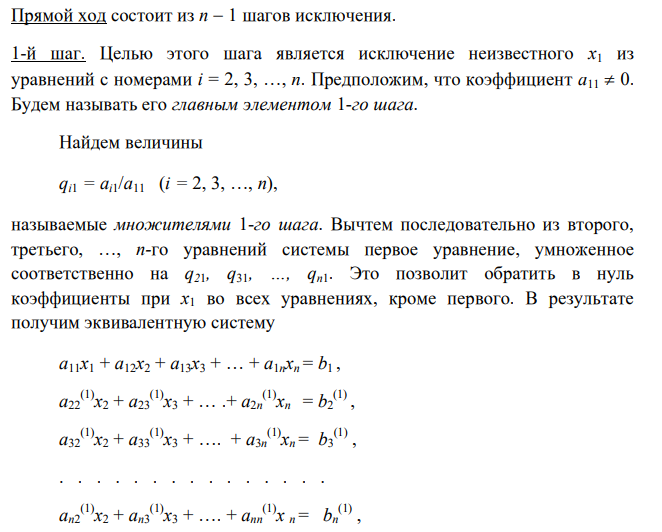
**2. КРАТКИЕ ТЕОРЕТИЧЕСКИЕ СВЕДЕНИЯ**

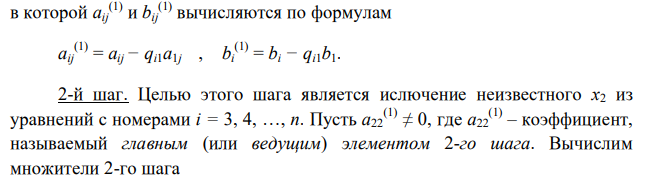
Метод Гаусса прекрасно подходит для решения систем линейных алгебраических уравнений (СЛАУ). Он обладает рядом преимуществ по сравнению с другими методами:

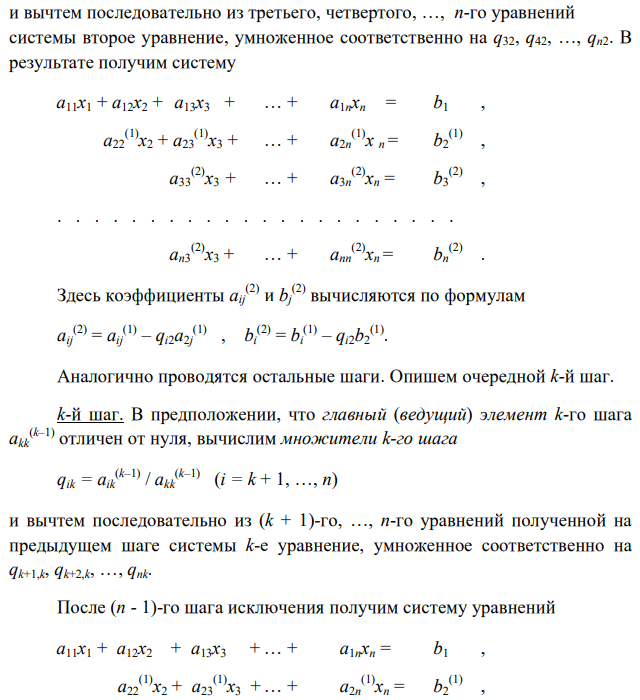
* во-первых, нет необходимости предварительно исследовать систему уравнений на совместность;
* во-вторых, методом Гаусса можно решать не только СЛАУ, в которых число уравнений совпадает с количеством неизвестных переменных и основная матрица системы невырожденная, но и системы уравнений, в которых число уравнений не совпадает с количеством неизвестных переменных или определитель основной матрицы равен нулю;
* в-третьих, метод Гаусса приводит к результату при сравнительно небольшом количестве вычислительных операций.

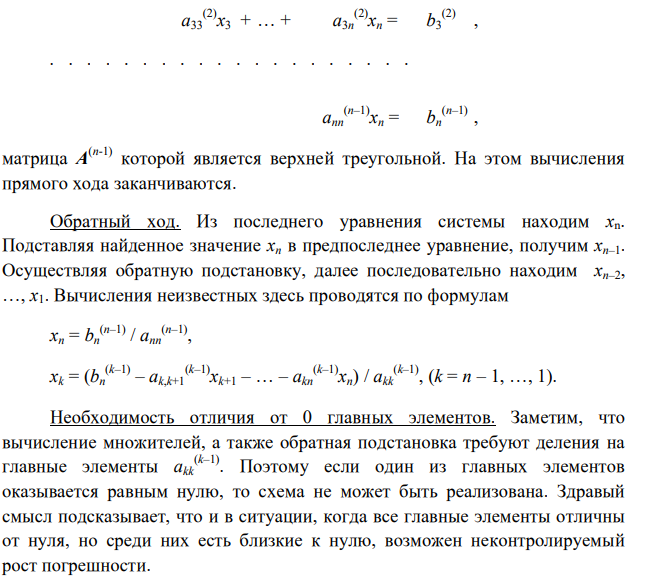
Метод Гаусса включает в себя прямой (приведение расширенной матрицы к ступенчатому виду, то есть получение нулей под главной диагональю) и обратный (получение нулей над главной диагональю расширенной матрицы) ходы. Прямой ход и называется методом Гаусса, обратный - методом Гаусса-Жордана, который отличается от первого только последовательностью исключения переменных.

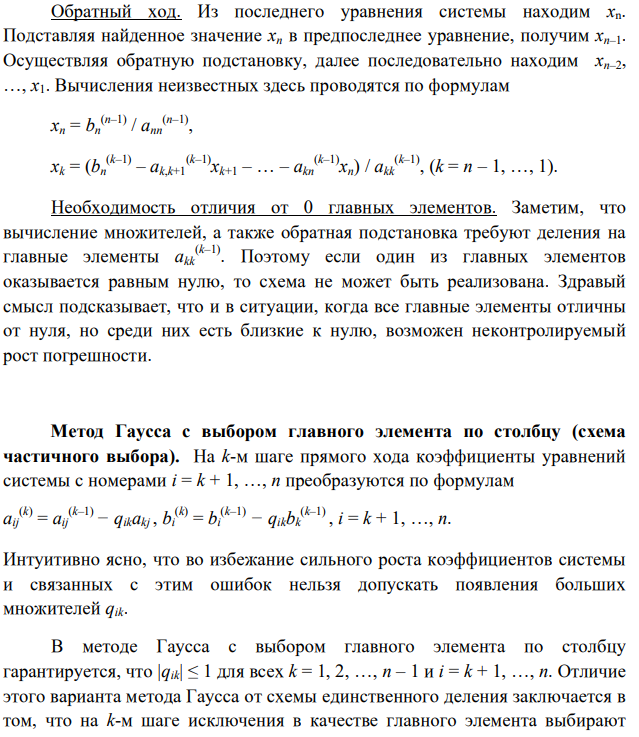
Метод Гаусса идеально подходит для решения систем содержащих больше трех линейных уравнений, для решения систем уравнений, которые не являются квадратными (чего не скажешь про метод Крамера и матричный метод). То есть метод Гаусса - наиболее универсальный метод для нахождения решения любой системы линейных уравнений, он работает в случае, когда система имеет бесконечно много решений или*.*

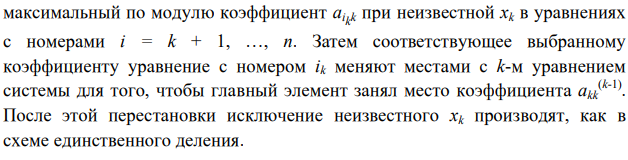


 qi2 = ai2(1) / a22(1) (i = 3, 4, …, n)









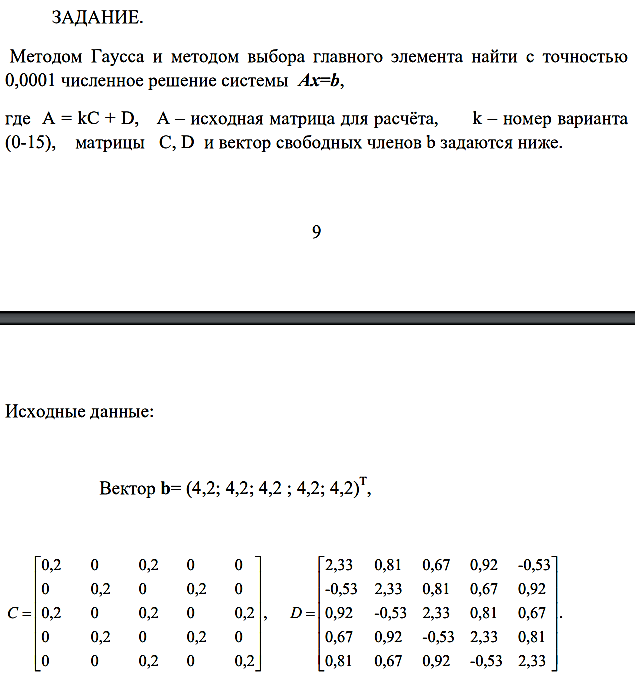
**Метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора).** В этой схеме допускается нарушение естественного порядка исключения неизвестных. На 1-м шаге метода среди элементов аij определяют максимальный по модулю элемент аij. Первое уравнение системы и уравнение с номером іi меняют местами. Далее стандартным образом производят исключение неизвестного хi из всех уравнений, кроме первого.

На *k*-м шаге метода среди коэффициентов aij(k-1) при неизвестных в уравнениях системы с номерами i = k, ..., n и выбирают максимальный по модулю коэффициент aij(k-1) . Затем *k-*е уравнение и уравнение, содержащее найденный коэффициент, меняют местами и исключают неизвестное xjk из уравнений с номерами i = k + 1. ..., n.

На этапе обратного хода неизвестные вычисляют в следующем порядке: хjn **,** хjn-1 **, …,** хj1 **.**

# **3. ЗАДАНИЕ** Методом Гаусса и методом выбора главного элемента найти с точностью 0,0001 численное решение системы ***Аx = b,*** где A = k\*C + D, A – исходная матрица для расчёта, k – номер варианта (0-15), матрицы C, D и вектор свободных членов b задаются ниже.

Исходные данные:

 **4. ПРОГРАММНАЯ РЕАЛИЗАЦИЯ**

#-------------главный исполняемый файл------------

import initial\_data as init

import gaussian\_elimination\_classic as gec

import gaussian\_elimination\_col\_mod as gecm

import gaussian\_elimination\_full\_matrix\_mod as gefm

import numpy as np

#M = init.test1()

#M = init.test2()

M = init.test3()

M = gec.forward\_move(M, (lambda x, y, z: None))

c\_solution = gec.backward\_move(M)

print(np.round(c\_solution.astype(float), 4))

M = init.get\_initial\_matrix()

M = gec.forward\_move(M, (lambda x, y, z: None))

c\_solution = gec.backward\_move(M)

print(np.round(c\_solution.astype(float), 4))

M = init.get\_initial\_matrix()

M = gec.forward\_move(M, gecm.col\_mod)

c\_solution = gec.backward\_move(M)

print(np.round(c\_solution.astype(float), 4))

M = init.get\_initial\_matrix()

arr = [i for i in range(M.shape[1] - 1)]

M = gec.forward\_move(M, gefm.full\_mod, arr)

c\_solution = gec.backward\_move(M, arr)

print(np.round(c\_solution.astype(float), 4))

#------------файл инициализации матрицы----------------

from fractions import Fraction as fr

import numpy as np

C = np.array([

[fr(0.2), 0, fr(0.2), 0, 0],

[0, fr(0.2), 0, fr(0.2), 0],

[fr(0.2), 0, fr(0.2), 0, fr(0.2)],

[0, fr(0.2), 0, fr(0.2), 0],

[fr(0.2), 0, fr(0.2), 0, fr(0.2)]

])

D = np.array([

[fr(2.33), fr(0.81), fr(0.67), fr(0.92), fr(-0.53)],

[fr(-0.53), fr(2.33), fr(0.81), fr(0.67), fr(0.92)],

[fr(0.92), fr(-0.53), fr(2.33), fr(0.81), fr(0.67)],

[fr(0.67), fr(0.92), fr(-0.53), fr(2.33), fr(0.81)],

[fr(0.81), fr(0.67), fr(0.92), fr(-0.53), fr(2.33)]

])

b = np.array([

[fr(4.2)],

[fr(4.2)],

[fr(4.2)],

[fr(4.2)],

[fr(4.2)]

])

k = 5

def test1():

M = np.array([

[fr(2),fr(3)],

[fr(4),fr(6)]

])

B = np.array([

[fr(7)],

[fr(14)]

])

col = M.shape[0]

M = np.transpose(M)

for c in range(col):

if all(i == 0 for i in M[c]):

M = np.delete(M, c, axis = 0)

M = np.transpose(M)

M\_ex = np.hstack((M,B))

if np.linalg.matrix\_rank(M.astype(float)) != np.linalg.matrix\_rank(M\_ex.astype(float)):

print("the matrix is inconsistent")

exit(0)

if np.linalg.matrix\_rank(M.astype(float)) != len(M[0]):

print("the matrix has an infinite number of solutions")

exit(0)

return M\_ex

def test2():

M = np.array([

[2,3],

[4,6]

])

B = np.array([

[7],

[12]

])

col = M.shape[0]

M = np.transpose(M)

for c in range(col):

if all(i == 0 for i in M[c]):

M = np.delete(M, c, axis = 0)

M = np.transpose(M)

M\_ex = np.hstack((M,B))

if np.linalg.matrix\_rank(M.astype(float)) != np.linalg.matrix\_rank(M\_ex.astype(float)):

print("the matrix is inconsistent")

exit(0)

if np.linalg.matrix\_rank(M.astype(float)) != len(M[0]):

print("the matrix has an infinite number of solutions")

exit(0)

return M\_ex

def test3():

M = np.array([

[2.,3.],

[4.,-5.]

])

B = np.array([

[7.],

[-6.]

])

col = M.shape[0]

M = np.transpose(M)

for c in range(col):

if all(i == 0 for i in M[c]):

M = np.delete(M, c, axis = 0)

M = np.transpose(M)

M\_ex = np.hstack((M,B))

if np.linalg.matrix\_rank(M.astype(float)) != np.linalg.matrix\_rank(M\_ex.astype(float)):

print("the matrix is inconsistent")

exit(0)

if np.linalg.matrix\_rank(M.astype(float)) != len(M[0]):

print("the matrix has an infinite number of solutions")

exit(0)

return M\_ex

def get\_initial\_matrix():

M = k\*C + D

col = M.shape[0]

M = np.transpose(M)

for c in range(col):

if all(i == 0 for i in M[c]):

M = np.delete(M, c, axis = 0)

M = np.transpose(M)

M\_ex = np.hstack((M,b))

if np.linalg.matrix\_rank(M.astype(float)) != np.linalg.matrix\_rank(M\_ex.astype(float)):

print("the matrix is inconsistent")

exit(0)

if np.linalg.matrix\_rank(M.astype(float)) != len(M[0]):

print("the matrix has an infinite number of solutions")

exit(0)

return M\_ex

#файл реализации метода Гаусса с передаваемой в него функцией модификации

from fractions import Fraction as fr

import numpy as np

def forward\_move(M, mod\_func, arr = None):

row, col = M.shape

for cur\_col in range(col - 1):

mod\_func(M, cur\_col, arr)

if M[cur\_col][cur\_col] == 0:

for r in range(cur\_col, row):

if M[r][cur\_col] != 0:

M[r], M[cur\_col] = M[cur\_col].copy(), M[r].copy()

break

deviser = M[cur\_col][cur\_col]

for r in range(cur\_col + 1, row):

M[r] -= M[cur\_col]\*M[r][cur\_col] / deviser

return M

def backward\_move(M, arr = None):

row, col = M.shape

solutions = np.zeros(col, dtype=fr)

for r in range(row - 1, -1, -1):

obtained = np.dot(M[r, r + 1:], solutions[r + 1:])

a = fr(M[r][col - 1] - obtained)

solutions[r] = a/M[r][r]

if arr:

resort(solutions, arr)

return solutions[:col - 1]

def resort(solutions, arr):

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

swap = arr.index(i)

a, b, c, d = solutions[swap], solutions[i], arr[swap], arr[i]

solutions[i], solutions[swap], arr[i], arr[swap] = a, b, c, d

#------------файл модификации выбора по столбцу--------------

from fractions import Fraction as fr

import numpy as np

def col\_mod(M, cur\_col, arr):

pivot = np.argmax(np.abs(M[cur\_col:, cur\_col])) + cur\_col

M[pivot], M[cur\_col] = M[cur\_col].copy(), M[pivot].copy()

#-----------файл модификации полного выбора------------------

from fractions import Fraction as fr

import numpy as np

import copy

def full\_mod(M, cur\_col, arr = None):

row, col = M.shape

mr, mc = cur\_col, cur\_col

max = np.abs(M[cur\_col][cur\_col])

for r in range(cur\_col, row):

for c in range(cur\_col, col - 1):

if np.abs(M[r][c]) > max:

max = np.abs(M[r][c])

mr, mc = r,c

arr[cur\_col], arr[mc] = arr[mc], arr[cur\_col]

M[mr], M[cur\_col] = M[cur\_col].copy(), M[mr].copy()

M[:, [cur\_col, mc]] = M[:, [mc, cur\_col]]

# **ПОЛУЧЕННЫЕ РЕЗУЛЬТАТЫ**

Тестовая матрица 1:



Вывод программного продукта:



Тестовая матрица 2:



Вывод программного продукта:



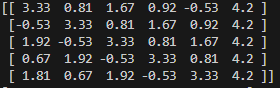
Тестовая матрица 3:



Вывод программного продукта:



Матрица варианта 5, составленная в соответствии с условием



Значения переменных, полученные после использования классического

метода Гаусса:



Значения переменных, полученные после использования схемы

частичного выбора:



Значения переменных, полученные после использования схемы полного выбора:



# **ОЦЕНКА**

Подсчет абсолютной погрешности приближенного решения:

0.5748 0.57479370 = 0. 00000630

Относительная погрешность приближенного числа:

# **ВЫВОДЫ**

Таким образом, в ходе выполнения данной лабораторной работы был применён метод Гаусса, метод Гаусса с выбором главного элемента по столбцу (схема частичного выбора) и метод Гаусса с выбором главного элемента по всей матрице (схема полного выбора) для решения системы линейных уравнений, составлены алгоритмы и созданы реализации соответствующих программ на языке Python для решения поставленной задачи, также проведена оценка.

Итак, метод Гаусса применим к любой системе линейных уравнений,

он идеально подходит для решения систем, содержащих больше трех

линейных уравнений. Метод Гаусса решения СЛАУ с числовыми

коэффициентами в силу простоты и однотипности выполняемых операций

пригоден для счета на электронно-вычислительных машинах.

Достоинства метода:

1. Менее трудоёмкий по сравнению с другими методами;
2. Позволяет однозначно установить, совместна система или нет, и если совместна, найти её решение;
3. Позволяет найти максимальное число линейно независимых

уравнений – ранг матрицы системы.