**МИНОБРНАУКИ РОССИИ**

**Санкт-Петербургский государственный**

**электротехнический университет**

**«ЛЭТИ» им. В.И. Ульянова (Ленина)**

**Кафедра МО ЭВМ**

отчет

**по практической работе №6**

**по дисциплине «Вычислительная математика»**

**Тема: Исследование обусловленности задачи решения систем линейных уравнений.**

|  |  |  |
| --- | --- | --- |
| Студентка гр. 1304 |  | Чернякова В.А. |
| Преподаватель |  | Попова Е.В. |

Санкт-Петербург

2022

## Цель работы.

## Изучение стандартной обусловленности задач решения систем линейных уравнений при различных вариантах неточных входных данных.

## Основные теоретические положения.

Рассматривается система линейных уравнений n-го порядка с вещественными коэффициентами (1)

В матричной форме записи эта система принимает вид (2)

, (2)

где – квадратная матрица коэффициентов системы, – вектор решений системы, – вектор свободных членов. Матрица – невырожденная, тогда решение системы (1) существует, единственно и устойчиво по входным данным. Это означает, что задача нахождения вектора – корректна.

Пусть – приближенное решение системы, тогда называется вектором погрешности системы, необходимо стремиться к его уменьшению. Возможно рассматривать критерий малости вектора который называется невязкой системы. Эти вектора связаны **.**

Удобной количественной характеристикой вектора является норма вектора. В вычислительной математике используются следующие три нормы (3)

(3)

За норму матрицы принимают максимальную величину, на которую преобразование, описываемое матрицей, может растянуть любой ненулевой вектор в выбранной норме . Векторным нормам подчинены следующие нормы матрицы (4)

где – собственные числа матрицы Задача вычисления вектора может быть плохо или хорошо обусловлена.

**Обусловленность задачи решения систем линейных алгебраических уравнений**

Рассмотрим случай, когда элементы матрицы заданы точно, а вектор-столбец свободных членов – приближенно. Оценки для абсолютной и относительной погрешности (5)

где - абсолютное число обусловленности, а - относительное число обусловленности (естественное число обусловленности). Максимальное естественное число обусловленности (6)

(6)

называют стандартным числом обусловленности.

Если элементы матрицы заданы приближенно и равны , а вектор-столбец свободных членов – точно, тогда оценка относительной погрешности (7)

(7)

где и .

Если с погрешностью заданы как коэффициенты матрицы, так и элементы вектора свободных членов, то справедливо неравенство (8)

(8)

**Использование wxMaxima для подсчета обратной матрицы**

Матрица – невырожденная, следовательно существует единственная обратная матрица . Для ее подсчета используется свободно распространяемый пакет системы компьютерной алгебры wxMaxima. Входная матрица задаётся с помощью выражения **matrix**(*стр1, стр2, ... стрN*), а обратная получается с помощью функции **invert**(*M*) (рисунок 1)

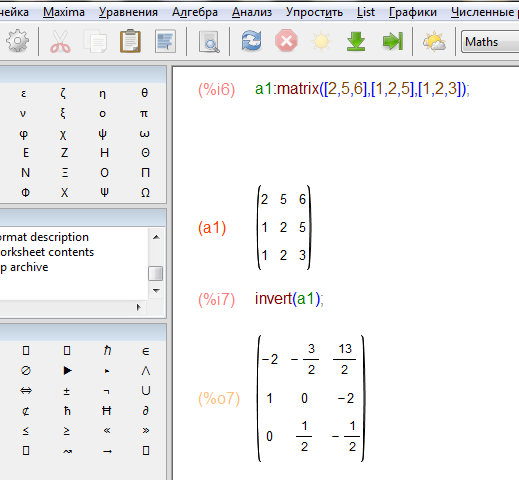


Рисунок 1 – Вычисление обратной матрицы с помощью функции *invert*

## Этапы выполнения практической работы.

1. Составить подпрограмму для решения системы линейных уравнений методом Гаусса и методом обратной матрицы.
2. Решить систему, подсчитать стандартное число обусловленности, используя тип данных с двойной точностью. Подсчет обратной матрицы производить с помощью системы компьютерной алгебры wxMaxima.
3. С помощью встроенной функции – генератора случайных чисел, добавить ошибки в вектор свободных членов. Найти решение новой системы, стандартное число обусловленности (6) и оценку стандартного числа обусловленности (7).
4. С помощью встроенной функции – генератора случайных чисел, добавить ошибки в значения элементов матрицы. Найти решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.
5. Добавить ошибки в значения элементов матрицы и вектора свободных членов. Найти решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.
6. Сделать выводы по полученным значениям.

## Варианты заданий практической работы.

Варианты заданий соответствуют списку группы. Первая матрица получается из матрицы варианта путем добавления столбца и строки так, чтобы новая матрица размерности 4 на 4 была невырожденной.

Другая матрица получается из новой заменой 2 строк (для четных номеров) и 2 столбцов (для нечетных номеров) на соответствующие элементы матрицы Гильберта ().

## Выполнение практической работы.

**Вариант 28(2).**

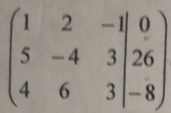
****

Рисунок 1 – исходная матрица.

1. Получаем первую матрицу из матрицы варианта путем добавления столбца и строки так, чтобы новая матрица размерности 4 на 4 была невырожденной. То есть ее определитель не равен 0.

Определитель полученной матрицы 286.

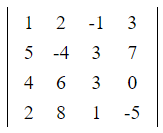


Рисунок 2 – первая матрица.

1. Вторую матрицу получим из новой заменой 2 строк (для четных номеров) на соответствующие элементы матрицы Гильберта ().

Заменим 2 и 4 строки.

Определитель данной матрицы -1501/4200.

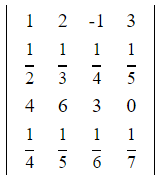


Рисунок 3 – вторая матрица.

Для 1 и 2 пункта изменим данный в варианте вектор свободных челнов: (0, 26, -8, 3)

1. Составить подпрограмму для решения системы линейных уравнений методом Гаусса и методом обратной матрицы.

**Функция GaussMethod.** Принимает на вход матрицу и вектор свободных членов. Используя теоретические положения о решении СЛУ методом Гаусса, с его помощью внутри функции находятся решения. Возвращает решение системы уравнений - вектор.

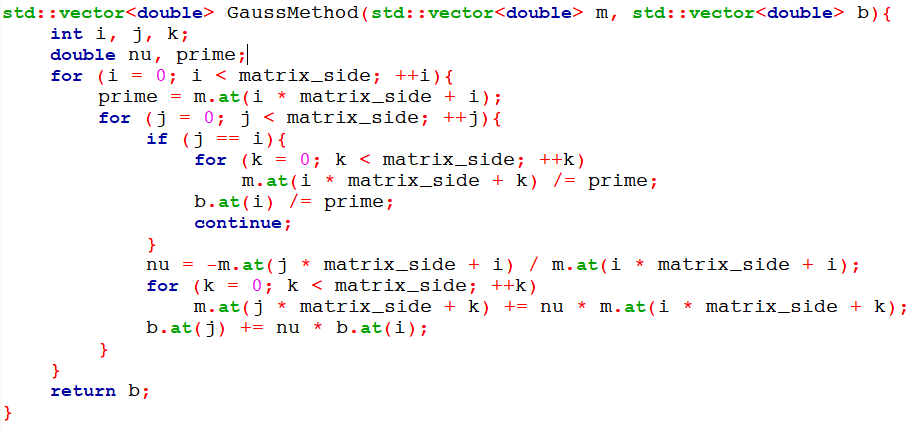
****

Рисунок 4 – функция GaussMethod, решающая СЛУ методом Гаусса.

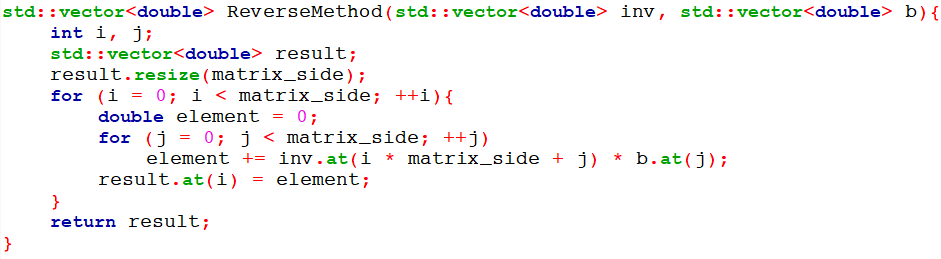
**Функция ReverseMethod.** Принимает на вход матрицу и вектор свободных членов. Используя теоретические положения о решении СЛУ методом обратной матрицы, с его помощью внутри функции находятся решения. Возвращает решение системы уравнений - вектор.

Рисунок 5 – функция ReverseMethod, решающая СЛУ методом обратной матрицы.

**Функция getInverseMatrix.** Так как для решения СЛУ необходима обратная матрица, была написана функция для ее получения.

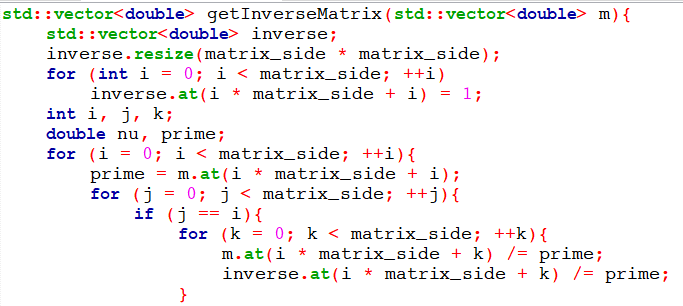
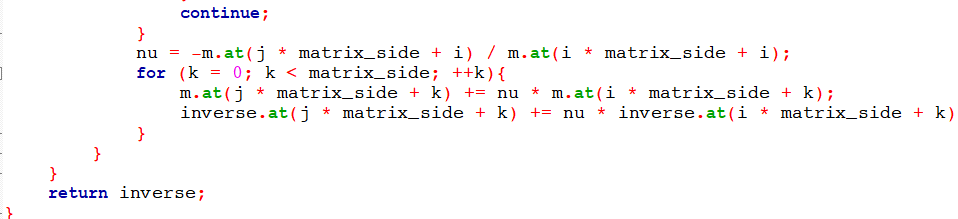


Рисунок 6 – функция getInverseMatrix получения обратной матрицы.

Выбранная норма у векторов - ||x|| с индексом бесконечность.

Выбранная норма у матрицы - ||А|| с индексом бесконечность.

**Исследование с входной матрицей.**

1. Решить систему, подсчитать стандартное число обусловленности, используя тип данных с двойной точностью.

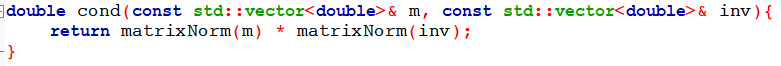
**Функция cond.** Функция нахождения стандартного числа обусловленности. Принимает исходную и обратную матрицу. Выполняются необходимые для нахождения числа обусловленности действия - перемножение норм исходной и обратной матрицы.

Рисунок 7 – функция cond нахождения стандартного числа обусловленности.

**Функция matrixNorm.** Так как для нахождения числа обусловленности необходимо знать норму матрицы, была написана данная функция. Она принимает на вход матрицу и находит ее норму.

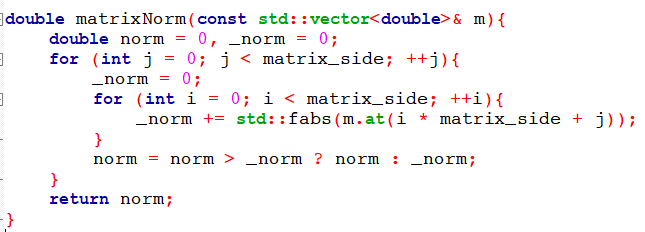


Рисунок 8 – функция matrixNorm нахождения нормы матрицы.

**Функция vectorNorm.** Так как для вычислений естественного числа обусловленности необходимо найти норму вектора, была создана данная функция.

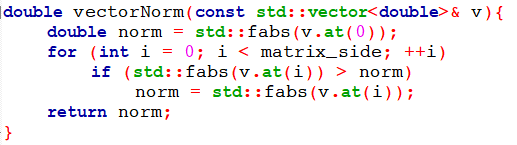


Рисунок 9 – функция vectorNorm нахождения нормы вектора.

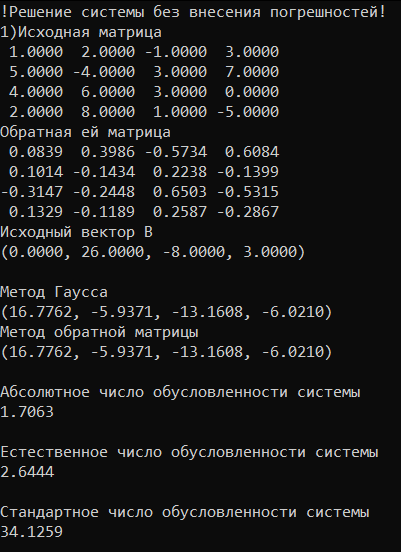


Рисунок 10 – решение системы и поиск числа обусловленности для исходной матрицы.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

1. С помощью встроенной функции – генератора случайных чисел, добавить ошибки в вектор свободных членов. Найти решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.

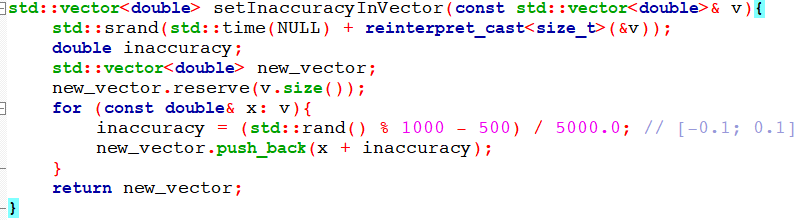
**Функция setInaccuracyInVector.** Данная функция принимает на вход вектор свободных членов и вносит в него ошибки с помощью встроенного генератора случайных чисел. Погрешность лежит в промежутке от -1.0 до 1.0.

Рисунок 11 – функция setInaccuracyInVector для внесения ошибок в вектор.

Для того, чтобы найти оценку стандартного числа обусловленности воспользуемся следующей формулой:

Так как ошибки вносятся в вектор.

Значение так же, как и

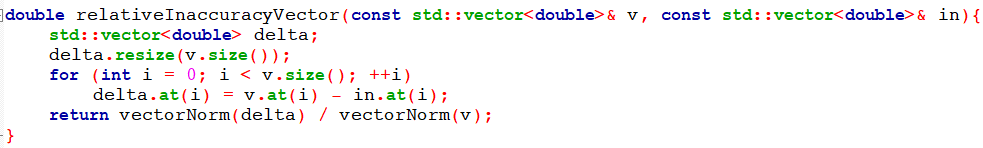
**Функция relativeInaccuracyVector.** Функция нахождения относительной погрешности векторов для осуществления оценки стандартного числа обусловленности. Принимает на вход два вектора – X\* и B\*.

Рисунок 12 – функция relativeInaccuracyVector нахождения относительной погрешности.

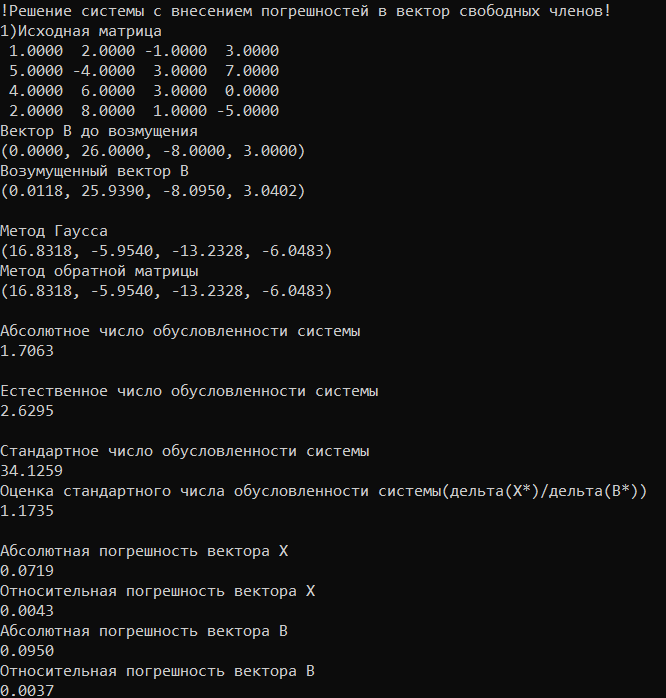


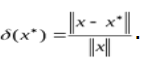
Рисунок 13 - решение новой системы (исходной), числа обусловленности и оценка стандартного числа обусловленности.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

Найдены абсолютные погрешности векторов X и B по формуле 

Найдены относительные погрешности векторов X и B по формуле 

Абсолютное и стандартное число обусловленности не изменились.

Абсолютная погрешность естественного числа обусловленности 0.0149, относительная – 0.0056.

Оценка:

δ(X\* )/δ(B\* ) = 1.1735

cond(A) = 34.1259

– Данное условие выполняется.

1. С помощью встроенной функции – генератора случайных чисел, добавим ошибки в значения элементов матрицы. Найдем решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.

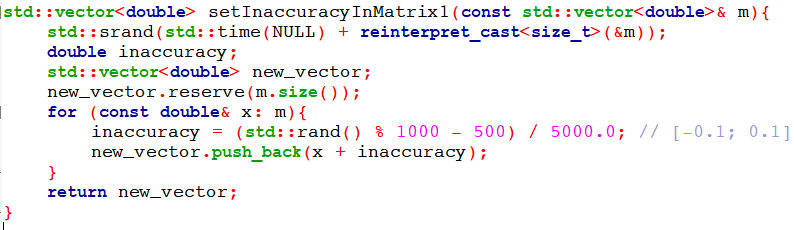
**Функция setInaccuracyInMatrix.** Данная функция принимает на вход матрицу и вносит в нее ошибки с помощью встроенного генератора случайных чисел. Погрешность лежит в промежутке от -1.0 до 1.0.

Рисунок 14 – функция setInaccuracyInMatrix для внесения ошибок в матрицу.

Для того, чтобы найти оценку стандартного числа обусловленности воспользуемся следующей формулой:

Так как ошибки вносятся в матрицу.

Где и .

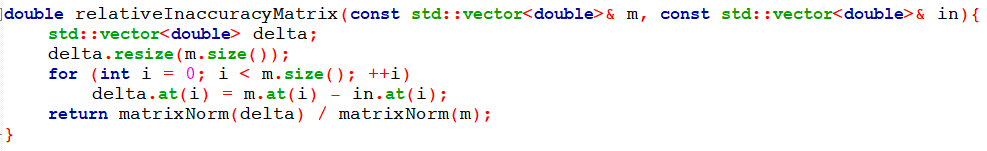
**Функция relativeInaccuracyMatrix.** Функция нахождения относительной погрешности матрицы для осуществления оценки стандартного числа обусловленности. Принимает на вход матрицу.

Рисунок 15 – функция relativeInaccuracyMatrix нахождения относительной погрешности.

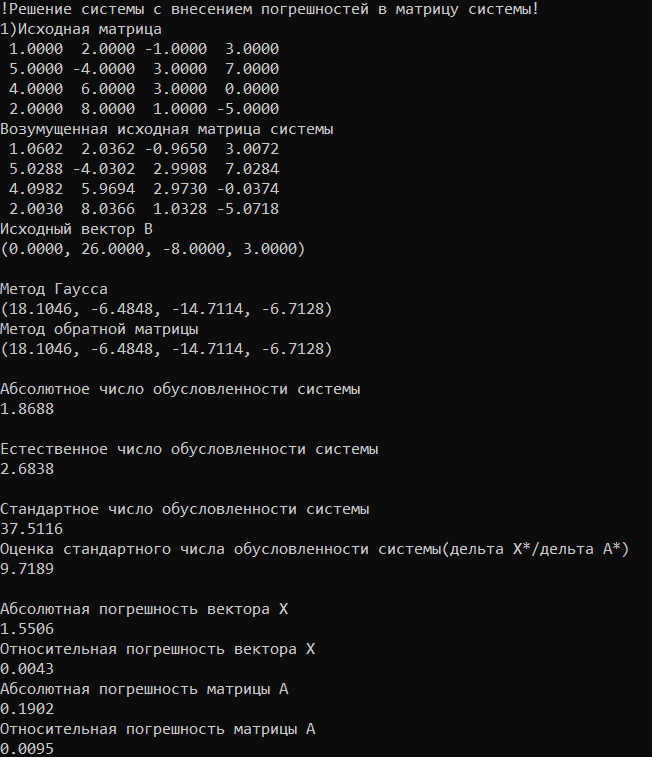


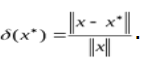
Рисунок 16 – решение новой системы (исходной), числа обусловленности и оценка стандартного числа обусловленности.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

Найдена абсолютная погрешность вектора X по формуле 

Найдена относительные погрешность вектора X по формуле 

Найдена абсолютная погрешность матрицы по формуле ∆(A\*) = ||A\*-A||.

Найдена относительная погрешность матрицы по формуле δ(A\* ) = ||A\*-A||/||A||.

Абсолютная и относительная погрешность абсолютного числа обусловленности 0.1625 и 0.9523 соответственно.

Абсолютная и относительная погрешность естественного числа обусловленности 0.0394 и 0.0149 соответственно.

Абсолютная и относительная погрешность стандартного числа обусловленности 3.3857 и 0.0992 соответственно.

Оценка:

δ(X\* )/ δ(A\* ) = 9.7189

cond(A) = 37.5116

δ(X\* )/ δ(A\* ) ≤cond(A) – данное условие выполняется.

1. Добавим ошибки в значения элементов матрицы и вектора свободных членов. Найдем решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.

Для того, чтобы найти оценку стандартного числа обусловленности воспользуемся следующей формулой:

Так как ошибки вносятся и в матрицу, и в вектор.

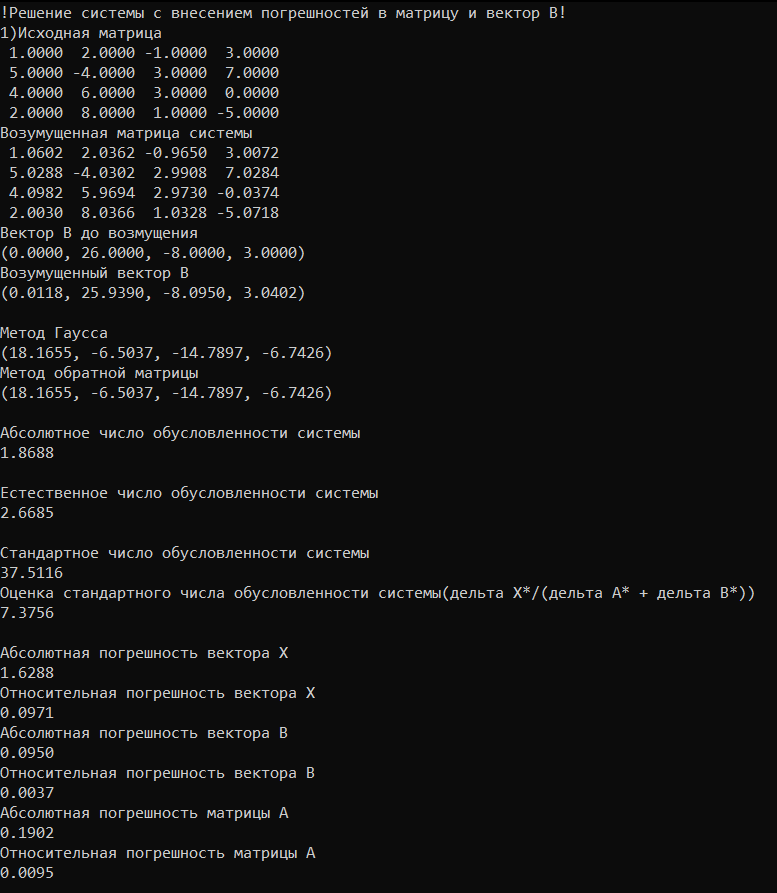


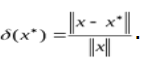
Рисунок 18 – решение новой системы (исходной), стандартное число обусловленности и оценка стандартного числа обусловленности.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

Найдены абсолютные погрешности векторов X и B по формуле 

Найдены относительные погрешности векторов X и B по формуле 

Найдена абсолютная погрешность матрицы по формуле ∆(A\*) = ||A\*-A||.

Найдена относительная погрешность матрицы по формуле δ(A\* ) = ||A\*-A||/||A||.

Абсолютная и относительная погрешность абсолютного числа обусловленности 0.1625 и 0.9523 соответственно.

Абсолютная и относительная погрешность естественного числа обусловленности 0.0241 и 0.0091 соответственно.

Абсолютная и относительная погрешность стандартного числа обусловленности 3.3857 и 0.0992 соответственно.

Оценка:

δ(X\* )/ (δ(A\* )+δ(B\* )) = 7.3756

cond(A) = 37.5116

δ(X\* )/ (δ(A\* )+δ(B\* )) ≤ cond(A) данное условие выполняется.

1. Вывод:

Для исходной матрицы. Стандартное число обусловленности системы 34.1259. При внесении ошибок в вектор свободных членов стандартное число 34.1259. Внося ошибки в значения элементов матрицы, получаем число обусловленности равно 37.5116. А при совмещении ошибок: и в вектор, и в матрицу – число обусловленности 37.5116. Таким образом, максимальное значение числа обусловленности достигается при внесении ошибок в значения элементов матрицы отдельно, так и при внесении ошибок сразу и в матрицу, и в вектор свободных членов. Минимальное значение получается при наличии матрицы и вектора без изменений, но и при возмущенном векторе тоже.

**Исследование с матрицей Гильберта.**

1. Решить систему, подсчитать стандартное число обусловленности, используя тип данных с двойной точностью.

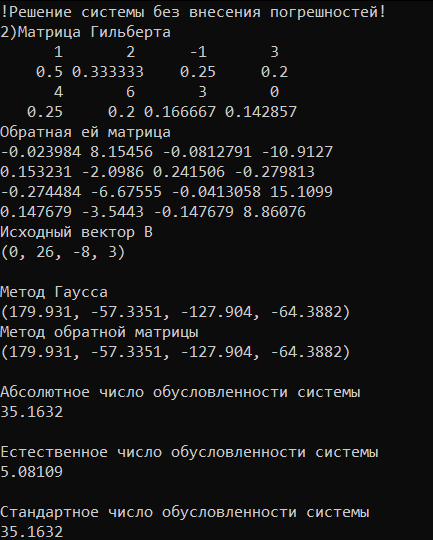


Рисунок 10 – решение системы и поиск числа обусловленности для исходной матрицы.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

1. С помощью встроенной функции – генератора случайных чисел, добавить ошибки в вектор свободных членов. Найти решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.

Для того, чтобы найти оценку стандартного числа обусловленности воспользуемся следующей формулой:

Так как ошибки вносятся в вектор.

Значение так же, как и

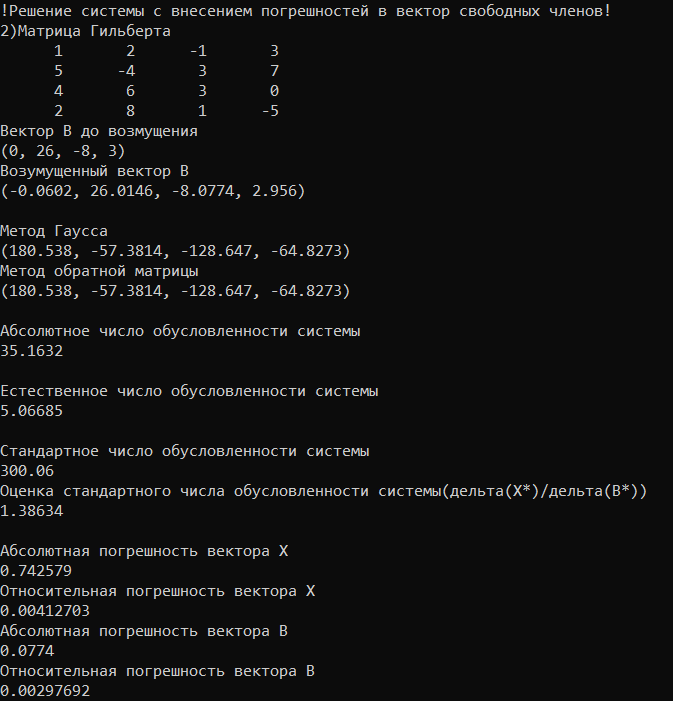


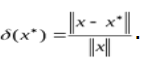
Рисунок 13 - решение новой системы (исходной), числа обусловленности и оценка стандартного числа обусловленности.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

Найдены абсолютные погрешности векторов X и B по формуле 

Найдены относительные погрешности векторов X и B по формуле 

Абсолютное число обусловленности не изменилось.

Абсолютная погрешность естественного числа обусловленности 0.01424, относительная – 0.0028.

Абсолютная погрешность стандартного числа обусловленности 264.8968, относительная – 7.5333.

Оценка:

δ(X\* )/δ(B\* ) = 1.38634

cond(A) = 300.06

– Данное условие выполняется.

1. С помощью встроенной функции – генератора случайных чисел, добавим ошибки в значения элементов матрицы. Найдем решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.

Для того, чтобы найти оценку стандартного числа обусловленности воспользуемся следующей формулой:

Так как ошибки вносятся в матрицу.

Где и .

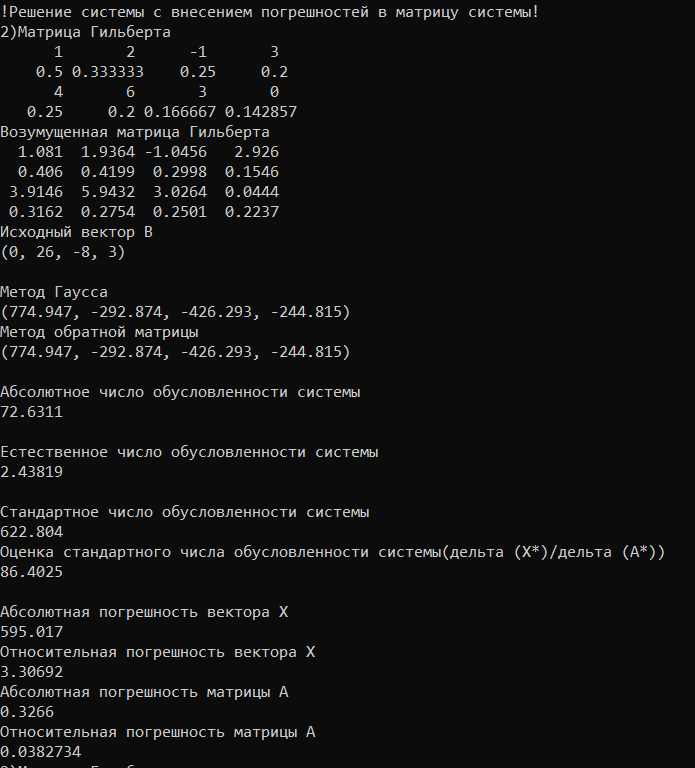


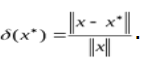
Рисунок 16 – решение новой системы (исходной), числа обусловленности и оценка стандартного числа обусловленности.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

Найдена абсолютная погрешность вектора X по формуле 

Найдена относительные погрешность вектора X по формуле 

Найдена абсолютная погрешность матрицы по формуле ∆(A\*) = ||A\*-A||.

Найдена относительная погрешность матрицы по формуле δ(A\* ) = ||A\*-A||/||A||.

Абсолютная и относительная погрешность абсолютного числа обусловленности 37.4679 и 1.0655 соответственно.

Абсолютная и относительная погрешность естественного числа обусловленности 2.6429 и 0.5201 соответственно.

Абсолютная и относительная погрешность стандартного числа обусловленности 587.6408 и 16.7118 соответственно.

Оценка:

δ(X\* )/ δ(A\* ) = 86.4025

cond(A) = 622.804

δ(X\* )/ δ(A\* ) ≤cond(A) – данное условие выполняется.

1. Добавим ошибки в значения элементов матрицы и вектора свободных членов. Найдем решение новой системы, стандартное число обусловленности и оценку стандартного числа обусловленности.

Для того, чтобы найти оценку стандартного числа обусловленности воспользуемся следующей формулой:

Так как ошибки вносятся и в матрицу, и в вектор.

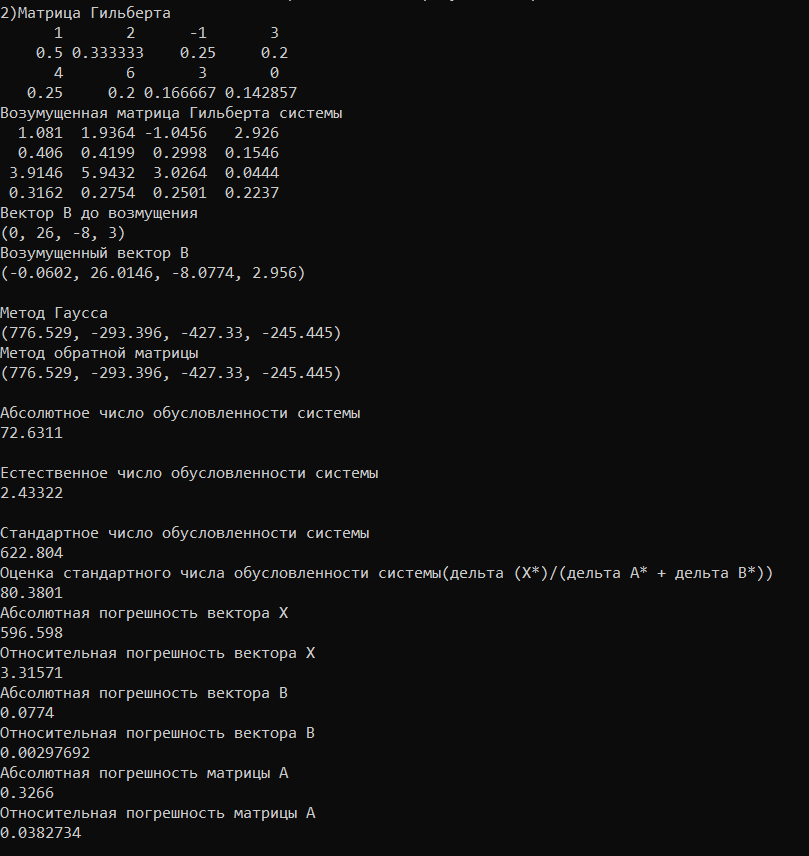


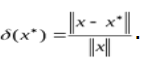
Рисунок 18 – решение новой системы (исходной), стандартное число обусловленности и оценка стандартного числа обусловленности.

Найдено абсолютное число обусловленности по формуле .

Найдено естественное число обусловленности по формуле .

Найдено стандартное число обусловленности по формуле .

Найдены абсолютные погрешности векторов X и B по формуле 

Найдены относительные погрешности векторов X и B по формуле 

Найдена абсолютная погрешность матрицы по формуле ∆(A\*) = ||A\*-A||.

Найдена относительная погрешность матрицы по формуле δ(A\* ) = ||A\*-A||/||A||.

Абсолютная и относительная погрешность абсолютного числа обусловленности 37.4679 и 1.0655 соответственно.

Абсолютная и относительная погрешность естественного числа обусловленности 2.63363 и 0.52112 соответственно.

Абсолютная и относительная погрешность стандартного числа обусловленности 587.6408 и 16.7118 соответственно.

Оценка:

δ(X\* )/ (δ(A\* )+δ(B\* )) =

cond(A) =

δ(X\* )/ (δ(A\* )+δ(B\* )) ≤ cond(A) данное условие выполняется.

1. Вывод:

Для матрицы Гильберта. Стандартное число обусловленности системы 703.2645. При внесении ошибок в вектор свободных членов стандартное число 300.0595. Внося ошибки в значения элементов матрицы, получаем число обусловленности равно 629.5705. А при совмещении ошибок: и в вектор, и в матрицу – число обусловленности 622.8042. Таким образом, максимальное значение числа обусловленности достигается, когда ошибки не вносятся. При возмущенном векторе значение стандартного числа обусловленности минимально. Значения числа обусловленности при внесении ошибок только в значения элементов матрицы, так и при возмущении вектора и матрицы одновременно, находятся между минимальным и максимальным значением.

**Общий вывод. Сравнение.**

Сравним начальные показатели матриц.

Входная матрица:

* Абсолютное число обусловленности 1.7063
* Естественное число 2.6444
* Стандартное число 34.1259

Матрица Гильберта:

* Абсолютное число обусловленности 35.1632
* Естественное число 5.08109
* Стандартное число 35.1632

Заметим, что числа обусловленности матрицы Гильберта больше обычной матрицы. Это говорит о том, что такая матрицы будет давать большие ошибки при решении системы.

Этот довод подтверждается, когда при внесении погрешностей в матрицу или и в матрицу и вектор свободных членов, погрешность вектора X достаточно большая.

Внесение погрешности в матрицу системы.

Входная матрица:

* Абсолютная погрешность вектора X 1.5506
* Относительная погрешность вектора X 0.0043

Матрица Гильберта:

* Абсолютная погрешность вектора X 595.017
* Относительная погрешность вектора X 3.30692

Внесение погрешности в матрицу системы и в вектор свободных членов.

Входная матрица:

* Абсолютная погрешность вектора X 1.6288
* Относительная погрешность вектора X 0.0971

Матрица Гильберта:

* Абсолютная погрешность вектора X 596.598
* Относительная погрешность вектора X 3.31517

Таким образом, модно сделать вывод, что матрица Гильберта является плохо обусловленной. То есть для такой матрицы решение системы уравнений практически является неустойчивым.

# ПРИЛОЖЕНИЕ.

#include <iostream>

#include <vector>

#include <cmath>

#include <ctime>

#include <cstdlib>

#include <iomanip>

const int matrix\_side = 4;

const std::vector<double> matrix = {1, 2, -1, 3,

5, -4, 3, 7,

4, 6, 3, 0,

2, 8, 1, -5};

const std::vector<double> gilbert\_matrix = {1, 2, -1, 3,

1/2.0, 1/3.0, 1/4.0, 1/5.0,

4, 6, 3, 0,

1/4.0, 1/5.0, 1/6.0, 1/7.0};

const std::vector<double> vectorB = {0, 26, -8, 3};

std::vector<double> GaussMethod(std::vector<double> m, std::vector<double> b){

int i, j, k;

double nu, prime;

for (i = 0; i < matrix\_side; ++i){

prime = m.at(i \* matrix\_side + i);

for (j = 0; j < matrix\_side; ++j){

if (j == i){

for (k = 0; k < matrix\_side; ++k)

m.at(i \* matrix\_side + k) /= prime;

b.at(i) /= prime;

continue;

}

nu = -m.at(j \* matrix\_side + i) / m.at(i \* matrix\_side + i);

for (k = 0; k < matrix\_side; ++k)

m.at(j \* matrix\_side + k) += nu \* m.at(i \* matrix\_side + k);

b.at(j) += nu \* b.at(i);

}

}

return b;

}

std::vector<double> getInverseMatrix(std::vector<double> m){//через единичную матрицу

std::vector<double> inverse;

inverse.resize(matrix\_side \* matrix\_side);

for (int i = 0; i < matrix\_side; ++i)

inverse.at(i \* matrix\_side + i) = 1;

int i, j, k;

double nu, prime;

for (i = 0; i < matrix\_side; ++i){

prime = m.at(i \* matrix\_side + i);

for (j = 0; j < matrix\_side; ++j){

if (j == i){

for (k = 0; k < matrix\_side; ++k){

m.at(i \* matrix\_side + k) /= prime;

inverse.at(i \* matrix\_side + k) /= prime;

}

continue;

}

nu = -m.at(j \* matrix\_side + i) / m.at(i \* matrix\_side + i);

for (k = 0; k < matrix\_side; ++k){

m.at(j \* matrix\_side + k) += nu \* m.at(i \* matrix\_side + k);

inverse.at(j \* matrix\_side + k) += nu \* inverse.at(i \* matrix\_side + k);

}

}

}

return inverse;

}

std::vector<double> ReverseMethod(std::vector<double> inv, std::vector<double> b){

int i, j;

std::vector<double> result;

result.resize(matrix\_side);

for (i = 0; i < matrix\_side; ++i){

double element = 0;

for (j = 0; j < matrix\_side; ++j)

element += inv.at(i \* matrix\_side + j) \* b.at(j);

result.at(i) = element;

}

return result;

}

double matrixNorm(const std::vector<double>& m){

double norm = 0, \_norm = 0;

for (int j = 0; j < matrix\_side; ++j){

\_norm = 0;

for (int i = 0; i < matrix\_side; ++i){

\_norm += std::fabs(m.at(i \* matrix\_side + j));

}

norm = norm > \_norm ? norm : \_norm;

}

return norm;

}

double cond(const std::vector<double>& m, const std::vector<double>& inv){

return matrixNorm(m) \* matrixNorm(inv);

}

std::vector<double> setInaccuracyInVector1(const std::vector<double>& v){

std::srand(std::time(NULL) + reinterpret\_cast<size\_t>(&v));

double inaccuracy;

std::vector<double> new\_vector;/\*

new\_vector.reserve(v.size());

for (const double& x: v){

inaccuracy = (std::rand() % 1000 - 500) / 5000.0; // [-0.1; 0.1]

new\_vector.push\_back(x + inaccuracy);

}\*/

new\_vector.push\_back(0.0118);

new\_vector.push\_back(25.9390);

new\_vector.push\_back(-8.0950);

new\_vector.push\_back(3.0402);

return new\_vector;

}

std::vector<double> setInaccuracyInVector2(const std::vector<double>& v){

std::srand(std::time(NULL) + reinterpret\_cast<size\_t>(&v));

double inaccuracy;

std::vector<double> new\_vector;/\*

new\_vector.reserve(v.size());

for (const double& x: v){

inaccuracy = (std::rand() % 1000 - 500) / 5000.0; // [-0.1; 0.1]

new\_vector.push\_back(x + inaccuracy);

}\*/

new\_vector.push\_back(-0.0602);

new\_vector.push\_back(26.0146);

new\_vector.push\_back(-8.0774);

new\_vector.push\_back(2.9560);

return new\_vector;

}

double vectorNorm(const std::vector<double>& v){

double norm = std::fabs(v.at(0));

for (int i = 0; i < matrix\_side; ++i)

if (std::fabs(v.at(i)) > norm)

norm = std::fabs(v.at(i));

return norm;

}

double AbsoluteAccuracyVector(const std::vector<double>& v, const std::vector<double>& in){

std::vector<double> delta;

delta.resize(v.size());

for (int i = 0; i < v.size(); ++i)

delta.at(i) = v.at(i) - in.at(i);

return vectorNorm(delta);

}

double relativeInaccuracyVector(const std::vector<double>& v, const std::vector<double>& in){

std::vector<double> delta;

delta.resize(v.size());

for (int i = 0; i < v.size(); ++i)

delta.at(i) = v.at(i) - in.at(i);

return vectorNorm(delta) / vectorNorm(v);

}

std::vector<double> setInaccuracyInMatrix1(const std::vector<double>& m){

std::srand(std::time(NULL) + reinterpret\_cast<size\_t>(&m));

double inaccuracy;

std::vector<double> new\_vector;/\*

new\_vector.reserve(m.size());

for (const double& x: m){

inaccuracy = (std::rand() % 1000 - 500) / 5000.0; // [-0.1; 0.1]

new\_vector.push\_back(x + inaccuracy);

}\*/

new\_vector.push\_back(1.0602);

new\_vector.push\_back(2.0362);

new\_vector.push\_back(-0.9650);

new\_vector.push\_back(3.0072);

new\_vector.push\_back(5.0288);

new\_vector.push\_back(-4.0302);

new\_vector.push\_back(2.9908);

new\_vector.push\_back(7.0284);

new\_vector.push\_back(4.0982);

new\_vector.push\_back(5.9694);

new\_vector.push\_back(2.9730);

new\_vector.push\_back(-0.0374);

new\_vector.push\_back(2.0030);

new\_vector.push\_back(8.0366);

new\_vector.push\_back(1.0328);

new\_vector.push\_back(-5.0718);

return new\_vector;

}

std::vector<double> setInaccuracyInMatrix2(const std::vector<double>& m){

std::srand(std::time(NULL) + reinterpret\_cast<size\_t>(&m));

double inaccuracy;

std::vector<double> new\_vector;/\*

new\_vector.reserve(m.size());

for (const double& x: m){

inaccuracy = (std::rand() % 1000 - 500) / 5000.0; // [-0.1; 0.1]

new\_vector.push\_back(x + inaccuracy);

}\*/

new\_vector.push\_back(1.0810);

new\_vector.push\_back(1.9364);

new\_vector.push\_back(-1.0456);

new\_vector.push\_back(2.9260);

new\_vector.push\_back(0.4060);

new\_vector.push\_back(0.4199);

new\_vector.push\_back(0.2998);

new\_vector.push\_back(0.1546);

new\_vector.push\_back(3.9146);

new\_vector.push\_back(5.9432);

new\_vector.push\_back(3.0264);

new\_vector.push\_back(0.0444);

new\_vector.push\_back(0.3162);

new\_vector.push\_back(0.2754);

new\_vector.push\_back(0.2501);

new\_vector.push\_back(0.2237);

return new\_vector;

}

double AbsoluteAccuracyMatrix(const std::vector<double>& m, const std::vector<double>& in){

std::vector<double> delta;

delta.resize(m.size());

for (int i = 0; i < m.size(); ++i)

delta.at(i) = m.at(i) - in.at(i);

return matrixNorm(delta);

}

double relativeInaccuracyMatrix(const std::vector<double>& m, const std::vector<double>& in){

std::vector<double> delta;

delta.resize(m.size());

for (int i = 0; i < m.size(); ++i)

delta.at(i) = m.at(i) - in.at(i);

return matrixNorm(delta) / matrixNorm(m);

}

void printMatrix(const std::vector<double>& m){

for (int i = 0; i < matrix\_side; ++i){

for (int j = 0; j < matrix\_side; ++j){

std::cout << std::setw(7);

std::cout << m.at(i \* matrix\_side + j) << " ";

}

std::cout << std::endl;

}

}

void printVector(const std::vector<double>& v){

std::cout << "(";

for (int i = 0; i < v.size() - 1; ++i)

std::cout << v.at(i) << ", ";

std::cout << v.at(v.size() - 1);

std::cout << ")" << std::endl;

}

int main(){

setlocale(LC\_ALL, "Russian");

//пункт 1

std::vector<double> solutionG = GaussMethod(matrix, vectorB);

std::vector<double> solutionI = ReverseMethod(getInverseMatrix(matrix), vectorB);

std::cout << std::fixed;

std::cout << std::setprecision(4);

std::cout << "!Решение системы без внесения погрешностей!" << std::endl;

std::cout << "1)Исходная матрица" << std::endl;

printMatrix(matrix);

std::cout << "Обратная ей матрица" << std::endl;

printMatrix(getInverseMatrix(matrix));

std::cout << "Исходный вектор B" << std::endl;

printVector(vectorB);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(solutionG);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(solutionI);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(matrix)) \* vectorNorm(vectorB) / vectorNorm(solutionG) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(matrix, getInverseMatrix(matrix)) << std::endl;

std::cout << std::endl;

std::vector<double> solutionG\_g = GaussMethod(gilbert\_matrix, vectorB);

std::vector<double> solutionI\_g = ReverseMethod(getInverseMatrix(gilbert\_matrix), vectorB);

std::cout << "2)Матрица Гильберта" << std::endl;

printMatrix(gilbert\_matrix);

std::cout << "Обратная ей матрица" << std::endl;

printMatrix(getInverseMatrix(gilbert\_matrix));

std::cout << "Исходный вектор B" << std::endl;

printVector(vectorB);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(solutionG\_g);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(solutionI\_g);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(gilbert\_matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(gilbert\_matrix)) \* vectorNorm(vectorB) / vectorNorm(solutionG\_g) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(gilbert\_matrix))<<std::endl;

//пункт 2

std::vector<double> in\_vectorB = setInaccuracyInVector1(vectorB);

std::vector<double> in2\_solutionG = GaussMethod(matrix, in\_vectorB);

std::vector<double> in2\_solutionI = ReverseMethod(getInverseMatrix(matrix), in\_vectorB);

std::cout << "!Решение системы с внесением погрешностей в вектор свободных членов!" << std::endl;

std::cout << "1)Исходная матрица" << std::endl;

printMatrix(matrix);

std::cout << "Вектор B до возмущения" << std::endl;

printVector(vectorB);

std::cout << "Возумущенный вектор B" << std::endl;

printVector(in\_vectorB);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(in2\_solutionG);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(in2\_solutionI);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(matrix)) \* vectorNorm(in\_vectorB) / vectorNorm(in2\_solutionG) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(matrix, getInverseMatrix(matrix)) << std::endl;

std::cout << "Оценка стандартного числа обусловленности системы(дельта(X\*)/дельта(B\*))" << std::endl;

std::cout << relativeInaccuracyVector(solutionG, in2\_solutionG) / relativeInaccuracyVector(vectorB, in\_vectorB) << std::endl;

std::cout << std::endl;

std::cout << "Абсолютная погрешность вектора X" << std::endl;

std::cout << AbsoluteAccuracyVector(solutionG, in2\_solutionG)<<std::endl;

std::cout << "Относительная погрешность вектора X" << std::endl;

std::cout << relativeInaccuracyVector(solutionG, in2\_solutionG)<<std::endl;

std::cout << "Абсолютная погрешность вектора B" << std::endl;

std::cout << AbsoluteAccuracyVector(vectorB, in\_vectorB)<<std::endl;

std::cout << "Относительная погрешность вектора B" << std::endl;

std::cout << relativeInaccuracyVector(vectorB, in\_vectorB)<<std::endl;

std::vector<double> in\_vectorB\_g = setInaccuracyInVector2(vectorB);

std::vector<double> in2\_solutionG\_g = GaussMethod(gilbert\_matrix, in\_vectorB\_g);

std::vector<double> in2\_solutionI\_g = ReverseMethod(getInverseMatrix(gilbert\_matrix), in\_vectorB\_g);

std::cout << "2)Матрица Гильберта" << std::endl;

printMatrix(matrix);

std::cout << "Вектор B до возмущения" << std::endl;

printVector(vectorB);

std::cout << "Возумущенный вектор B" << std::endl;

printVector(in\_vectorB\_g);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(in2\_solutionG\_g);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(in2\_solutionI\_g);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(gilbert\_matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(gilbert\_matrix)) \* vectorNorm(in\_vectorB\_g) / vectorNorm(in2\_solutionG\_g) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(gilbert\_matrix, getInverseMatrix(gilbert\_matrix)) << std::endl;

std::cout << "Оценка стандартного числа обусловленности системы(дельта(X\*)/дельта(B\*))" << std::endl;

std::cout << relativeInaccuracyVector(solutionG\_g, in2\_solutionG\_g) / relativeInaccuracyVector(vectorB, in\_vectorB\_g) << std::endl;

std::cout << std::endl;

std::cout << "Абсолютная погрешность вектора X" << std::endl;

std::cout << AbsoluteAccuracyVector(solutionG\_g, in2\_solutionG\_g)<<std::endl;

std::cout << "Относительная погрешность вектора X" << std::endl;

std::cout << relativeInaccuracyVector(solutionG\_g, in2\_solutionG\_g)<<std::endl;

std::cout << "Абсолютная погрешность вектора B" << std::endl;

std::cout << AbsoluteAccuracyVector(vectorB, in\_vectorB\_g)<<std::endl;

std::cout << "Относительная погрешность вектора B" << std::endl;

std::cout << relativeInaccuracyVector(vectorB, in\_vectorB\_g)<<std::endl;

//пункт 3

std::cout << "!Решение системы с внесением погрешностей в матрицу системы!" << std::endl;

std::vector<double> in\_matrix = setInaccuracyInMatrix1(matrix);

std::vector<double> in1\_solutionG = GaussMethod(in\_matrix, vectorB);

std::vector<double> in1\_solutionI = ReverseMethod(getInverseMatrix(in\_matrix), vectorB);

std::cout << "1)Исходная матрица"<<std::endl;

printMatrix(matrix);

std::cout << "Возумущенная исходная матрица системы" << std::endl;

printMatrix(in\_matrix);

std::cout << "Исходный вектор B" << std::endl;

printVector(vectorB);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(in1\_solutionG);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(in1\_solutionI);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix)) \* vectorNorm(vectorB) / vectorNorm(in1\_solutionG) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(in\_matrix, getInverseMatrix(in\_matrix)) << std::endl;

std::cout << "Оценка стандартного числа обусловленности системы(дельта Xдельта A\*)" << std::endl;

std::cout << relativeInaccuracyVector(solutionG, in1\_solutionG) / relativeInaccuracyMatrix(matrix, in\_matrix) << std::endl;

std::cout << std::endl;

std::cout << "Абсолютная погрешность вектора X" << std::endl;

std::cout << AbsoluteAccuracyVector(solutionG, in1\_solutionG)<<std::endl;

std::cout << "Относительная погрешность вектора X" << std::endl;

std::cout << relativeInaccuracyVector(solutionG, in2\_solutionG)<<std::endl;

std::cout << "Абсолютная погрешность матрицы A" << std::endl;

std::cout << AbsoluteAccuracyMatrix(matrix, in\_matrix)<<std::endl;

std::cout << "Относительная погрешность матрицы A" << std::endl;

std::cout << relativeInaccuracyMatrix(matrix, in\_matrix)<<std::endl;

std::vector<double> in\_matrix\_g = setInaccuracyInMatrix2(gilbert\_matrix);

std::vector<double> in1\_solutionG\_g = GaussMethod(in\_matrix\_g, vectorB);

std::vector<double> in1\_solutionI\_g = ReverseMethod(getInverseMatrix(in\_matrix\_g), vectorB);

std::cout << "2)Матрица Гильберта"<<std::endl;

printMatrix(gilbert\_matrix);

std::cout << "Возумущенная матрица Гильберта" << std::endl;

printMatrix(in\_matrix\_g);

std::cout << "Исходный вектор B" << std::endl;

printVector(vectorB);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(in1\_solutionG\_g);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(in1\_solutionI\_g);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix\_g)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix\_g)) \* vectorNorm(in\_vectorB\_g) / vectorNorm(in1\_solutionG\_g) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(in\_matrix\_g, getInverseMatrix(in\_matrix\_g)) << std::endl;

std::cout << "Оценка стандартного числа обусловленности системы(дельта (X\*)/дельта (A\*))" << std::endl;

std::cout << relativeInaccuracyVector(solutionG\_g, in1\_solutionG\_g) / relativeInaccuracyMatrix(gilbert\_matrix, in\_matrix\_g) << std::endl;

std::cout << std::endl;

std::cout << "Абсолютная погрешность вектора X" << std::endl;

std::cout << AbsoluteAccuracyVector(solutionG\_g, in1\_solutionG\_g)<<std::endl;

std::cout << "Относительная погрешность вектора X" << std::endl;

std::cout << relativeInaccuracyVector(solutionG\_g, in1\_solutionG\_g)<<std::endl;

std::cout << "Абсолютная погрешность матрицы A" << std::endl;

std::cout << AbsoluteAccuracyMatrix(gilbert\_matrix, in\_matrix\_g)<<std::endl;

std::cout << "Относительная погрешность матрицы A" << std::endl;

std::cout << relativeInaccuracyMatrix(gilbert\_matrix, in\_matrix\_g)<<std::endl;

//пункт 4

std::cout << "!Решение системы с внесением погрешностей в матрицу и вектор B!" << std::endl;

std::vector<double> in3\_solutionG = GaussMethod(in\_matrix, in\_vectorB);

std::vector<double> in3\_solutionI = ReverseMethod(getInverseMatrix(in\_matrix), in\_vectorB);

std::cout << "1)Исходная матрица"<<std::endl;

printMatrix(matrix);

std::cout << "Возумущенная матрица системы" << std::endl;

printMatrix(in\_matrix);

std::cout << "Вектор B до возмущения" << std::endl;

printVector(vectorB);

std::cout << "Возумущенный вектор B" << std::endl;

printVector(in\_vectorB);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(in3\_solutionG);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(in3\_solutionI);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix)) \* vectorNorm(in\_vectorB) / vectorNorm(in3\_solutionG) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(in\_matrix, getInverseMatrix(in\_matrix)) << std::endl;

std::cout << "Оценка стандартного числа обусловленности системы(дельта (X\*)/(дельта A\* + дельта B\*))" << std::endl;

std::cout << relativeInaccuracyVector(solutionG, in3\_solutionG) / (relativeInaccuracyVector(vectorB, in\_vectorB) +

relativeInaccuracyMatrix(matrix, in\_matrix)) << std::endl;

std::cout << std::endl;

std::cout << "Абсолютная погрешность вектора X" << std::endl;

std::cout << AbsoluteAccuracyVector(solutionG, in3\_solutionG)<<std::endl;

std::cout << "Относительная погрешность вектора X" << std::endl;

std::cout << relativeInaccuracyVector(solutionG, in3\_solutionG)<<std::endl;

std::cout << "Абсолютная погрешность вектора B" << std::endl;

std::cout << AbsoluteAccuracyVector(vectorB, in\_vectorB)<<std::endl;

std::cout << "Относительная погрешность вектора B" << std::endl;

std::cout << relativeInaccuracyVector(vectorB, in\_vectorB)<<std::endl;

std::cout << "Абсолютная погрешность матрицы A" << std::endl;

std::cout << AbsoluteAccuracyMatrix(matrix, in\_matrix)<<std::endl;

std::cout << "Относительная погрешность матрицы A" << std::endl;

std::cout << relativeInaccuracyMatrix(matrix, in\_matrix)<<std::endl;

std::vector<double> in3\_solutionG\_g = GaussMethod(in\_matrix\_g, in\_vectorB\_g);

std::vector<double> in3\_solutionI\_g = ReverseMethod(getInverseMatrix(in\_matrix\_g), in\_vectorB\_g);

std::cout << "2)Матрица Гильберта"<<std::endl;

printMatrix(gilbert\_matrix);

std::cout << "Возумущенная матрица Гильберта системы" << std::endl;

printMatrix(in\_matrix\_g);

std::cout << "Вектор B до возмущения" << std::endl;

printVector(vectorB);

std::cout << "Возумущенный вектор B" << std::endl;

printVector(in\_vectorB\_g);

std::cout << std::endl;

std::cout << "Метод Гаусса" << std::endl;

printVector(in3\_solutionG\_g);

std::cout << "Метод обратной матрицы" << std::endl;

printVector(in3\_solutionI\_g);

std::cout << std::endl;

std::cout << "Абсолютное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix\_g)) << std::endl;

std::cout << std::endl;

std::cout << "Естественное число обусловленности системы" << std::endl;

std::cout << matrixNorm(getInverseMatrix(in\_matrix\_g)) \* vectorNorm(in\_vectorB\_g) / vectorNorm(in3\_solutionG\_g) << std::endl;

std::cout << std::endl;

std::cout << "Стандартное число обусловленности системы" << std::endl;

std::cout << cond(in\_matrix\_g, getInverseMatrix(in\_matrix\_g)) << std::endl;

std::cout << "Оценка стандартного числа обусловленности системы(дельта (X\*)/(дельта A\* + дельта B\*))" << std::endl;

std::cout << relativeInaccuracyVector(solutionG\_g, in3\_solutionG\_g) / (relativeInaccuracyVector(vectorB, in\_vectorB\_g) +

relativeInaccuracyMatrix(gilbert\_matrix, in\_matrix\_g)) << std::endl;

std::cout << "Абсолютная погрешность вектора X" << std::endl;

std::cout << AbsoluteAccuracyVector(solutionG\_g, in3\_solutionG\_g)<<std::endl;

std::cout << "Относительная погрешность вектора X" << std::endl;

std::cout << relativeInaccuracyVector(solutionG\_g, in3\_solutionG\_g)<<std::endl;

std::cout << "Абсолютная погрешность вектора B" << std::endl;

std::cout << AbsoluteAccuracyVector(vectorB, in\_vectorB\_g)<<std::endl;

std::cout << "Относительная погрешность вектора B" << std::endl;

std::cout << relativeInaccuracyVector(vectorB, in\_vectorB\_g)<<std::endl;

std::cout << "Абсолютная погрешность матрицы A" << std::endl;

std::cout << AbsoluteAccuracyMatrix(gilbert\_matrix, in\_matrix\_g)<<std::endl;

std::cout << "Относительная погрешность матрицы A" << std::endl;

std::cout << relativeInaccuracyMatrix(gilbert\_matrix, in\_matrix\_g)<<std::endl;

std::cout << std::endl;

}